TPTeam™: A Collaborative Testing Tool Developed with the Eclipse Equinox and Communication Projects

Robert Patrick Brady

A Thesis in the Field of Information Technology

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

November 2007

© 2007 Robert P. Brady

Abstract

The aim of this thesis is to create an exemplary collaborative testing tool called TPTeam by using the Eclipse Equinox and Communication Framework (ECF) projects. This goal is achieved by the development of three components: TPBuddy™, a rich GUI client; TPBridge™, an event-driven plug-in that bridges communication across Java Virtual Machines (JVMs); and TPManager™, a server-side test management plug-in.

All three components utilize a novel service-oriented architecture derived from Equinox and distributed events to achieve collaborative unit testing. Out-of-JVM communication is done via instant messaging through Google Talk servers and the open Extensible Messaging and Presence Protocol (XMPP). Tests are executed in the form of Eclipse Test and Performance Tools Platform (TPTP) JUnit test suites, which allow for server-side management. The work represents a practical first step in utilizing newly available Eclipse open source projects and public infrastructure for the creation of collaborative tools.

Acknowledgments

I would like to thank my ALM in IT Thesis Director, Dr. Bill Robinson, for his help. He provided several valuable suggestions such as the use of object-relational mapping tools and validating the design with my own test suites. His advice was greatly appreciated.

My wife, Dr. Susan Michaud Brady (Harvard FAS Ph.D., ’93), offered emotional support and was understanding throughout the duration of the project. Without her backing, this project could not have been completed.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Agile methods require a high degree of real-time collaboration throughout the Application Lifecycle Management (ALM) process. Software development in this case can be represented as a continuum of activities as shown in Figure 1 below.

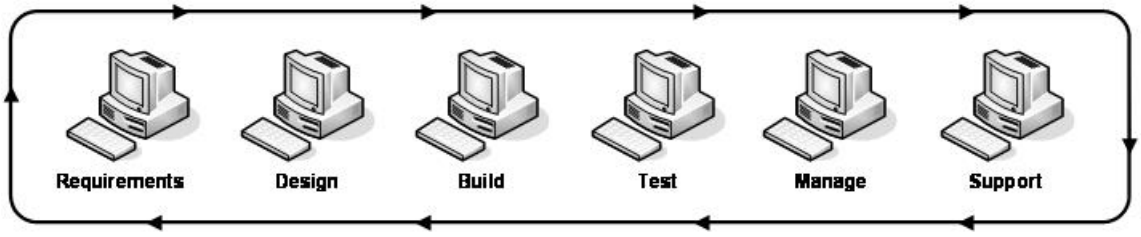

Figure 1 Application Lifecycle (Brady 2006)

The continuous ALM process requires not only collaboration between the different tools involved (e.g., between Test and Management), but intra-team (e.g., test practitioners "x" and "y") and also intra-group (e.g., between test practitioners, managers, and executives).

However, many existing development tools are designed essentially for a solo user. Studies have shown that distributed developers using real-time collaborative tools can complete some programming tasks at least twice as fast versus utilizing the solo user tool equivalents (Cook, Irwin, & Churcher, 2005).

Some of the major Java IDE providers have acknowledged the benefits of real-time collaborative tools and have taken steps to incorporate these features into their products. For example, the most widely used Java IDE, Eclipse (Zeichick, 2006), established the Eclipse Communication Framework (ECF) as an incubation project (The Eclipse Foundation, 2007a). ECF exposes an API for the sharing of Eclipse project resources across distributed workbenches as well as peer-to-peer messaging. Borland's commercial product JBuilder (Borland, 2007) provides "Virtual Peer Programming" with features such as: peer-to-peer chatting; shared projects, files, and stack traces; and conversation logging. Likewise Sun's commercially available Java Studio Enterprise (JSE) (Sun Microsystems, 2007a) supports peer instant messaging, shared files, and collaborative editing. The JBuilder and JSE products utilize coarse-grained interactions at the file level rather than the data structure or object level.

An existing Eclipse project utilizing ECF for tool integration and collaboration has recently been proposed and is undergoing incubation. The project is called Corona (The Eclipse Foundation, 2007b). It plans to provide tool integration by exposing Eclipse plug-in functionality between peers on different JVMs via Web services and ECF. However, the framework is complex and bloated with planned features many users may not incorporate into their projects. For example, the "basic" demonstration of the project utilizes approximately 900 of Corona developed classes for a simple CVS collaboration case. The thesis project by the author accomplishes a similar task with an order of magnitude fewer classes. Corona has a planned component that will not only utilize general Web services, but WS-DM or the Web services for Distributed Management specification (OASIS, 2007a). This example is an excellent characterization of Corona in

general. Rather than release simple and useful components in a timely manner, the Corona team promises exotic and complicated components that attract little if any potential users. Thus, rather than releasing a simple Web services bundle so that users could implement collaboration with SOAP over HTTP, Corona has released a complicated and cumbersome WS-DM package with little documentation. This lack of fundamental component delivery and third-party developer support is the result of over four years of private and public development effort.

There is a need for an Eclipse open source alternative to some of the previously mentioned commercial and open source solutions. This thesis provides not only an alternative framework that is lightweight, but also introduces an example tool from which others can follow a similar design pattern. The author presented preliminary details of this project at the international EclipseWorld 2006 conference in Cambridge, MA (Brady, 2006). This refereed acceptance is further evidence of the non-duplicative efforts and potential usefulness of this work.

Publicity and evangelism generated by the author should help motivate others to extend or add to the current work. The author, already an elected Eclipse committer, anticipates that the results of the current work would be incorporated into an Eclipse project, most likely ECF.

Overview of TPTeam

A typical TPTeam usage scenario would have the following actors present: a tester ("Tom Tester"), a project manager ("Polina PM"), a collaboration server, a

TPManager server, and a machine under test (MUT) where deployed TPTP JUnit test suites are located. This scenario is depicted below in Figure 2.

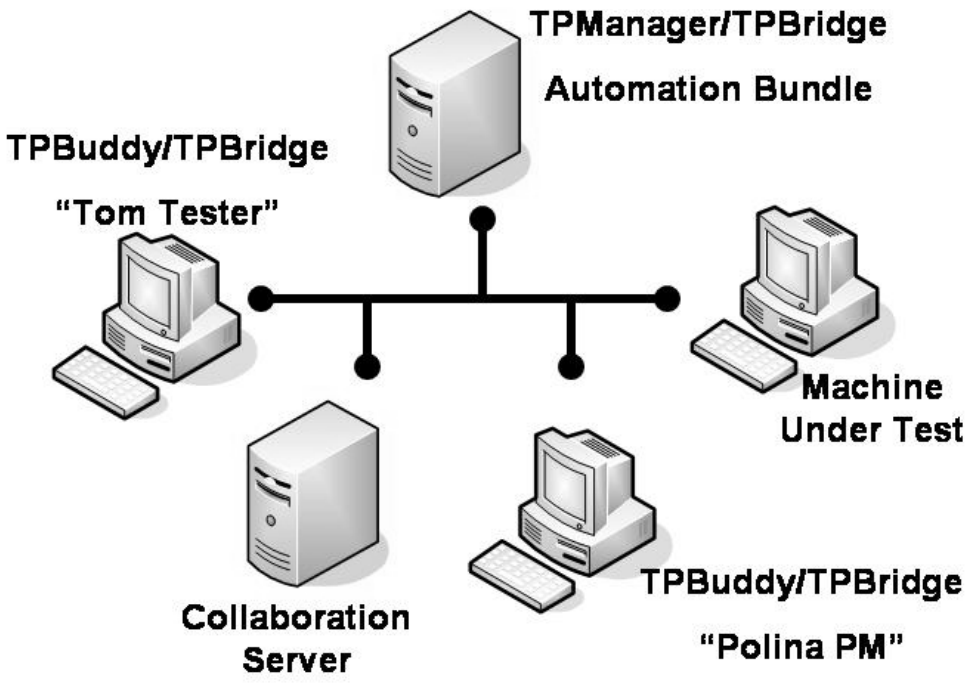

Figure 2 Typical TPTeam Scenario

The use case steps carried out would be:

1. Tom Tester requests a JUnit test execution with his TPBuddy GUI.
2. Tom's TPBuddy sends a test execution request ("testexecreq") event via its TPBridge plug-in.
3. The TPBridge plug-in utilizes the ECF libraries to forward the event to the Collaboration Server under the appropriate communications protocol.

4. The Collaboration Server receives the testexecreq event bundled inside a standard protocol message and delivers it through the network to the recipient, the TPManager instance.
5. The TPManager's TPBridge plug-in receives the testexecreq event and forwards it to TPManager's TPTP component.
6. TPManager's TPTP component begins to execute a TPTP test suite on the Machine Under Test (MUT).
7. TPManager collects the test results from the MUT, broadcasts a test execution response ("testexecresp") event containing the test result.
8. The Collaboration Server receives the testexecresp event and forwards it to the TPBridge components of all TPBuddies specified in the body of the testexecresp event.
9. All TPBuddy GUIs are updated in real time with the test result after receiving testexecresp event updates from their component TPBridge.
10. Polina PM notices new results in her TPBuddy GUI, and chats with Tom Tester about the results via the Instant Messenger (IM) interface of her TPBuddy GUI.

A screenshot of the TPBuddy Rich Client Platform (RCP) GUI under the above circumstances is given below in Figure 3. The Project Test Tree View is shown with its tab highlighted. The JUnit test TestTPEvent is highlighted within the test tree. Two test execution results have come into the view in real time and are displayed as children under the test node. The result verdict ("pass" in these cases) as well as execution time and ECF address of the person who requested the execution are summarized in the execution

results as well. Various other tabbed views also exist: Project View, which tabulates projects to which the user is assigned; Project Report View, a provider of charts that summarize project status; Test Details View, a display of detailed information about a selected test definition or folder; Roster View, shows a list of all other TPBuddies online and provides a right-clickable context menu for chatting with them; and finally, Event History View, a tabulated log of all TPTeam events received during a session.

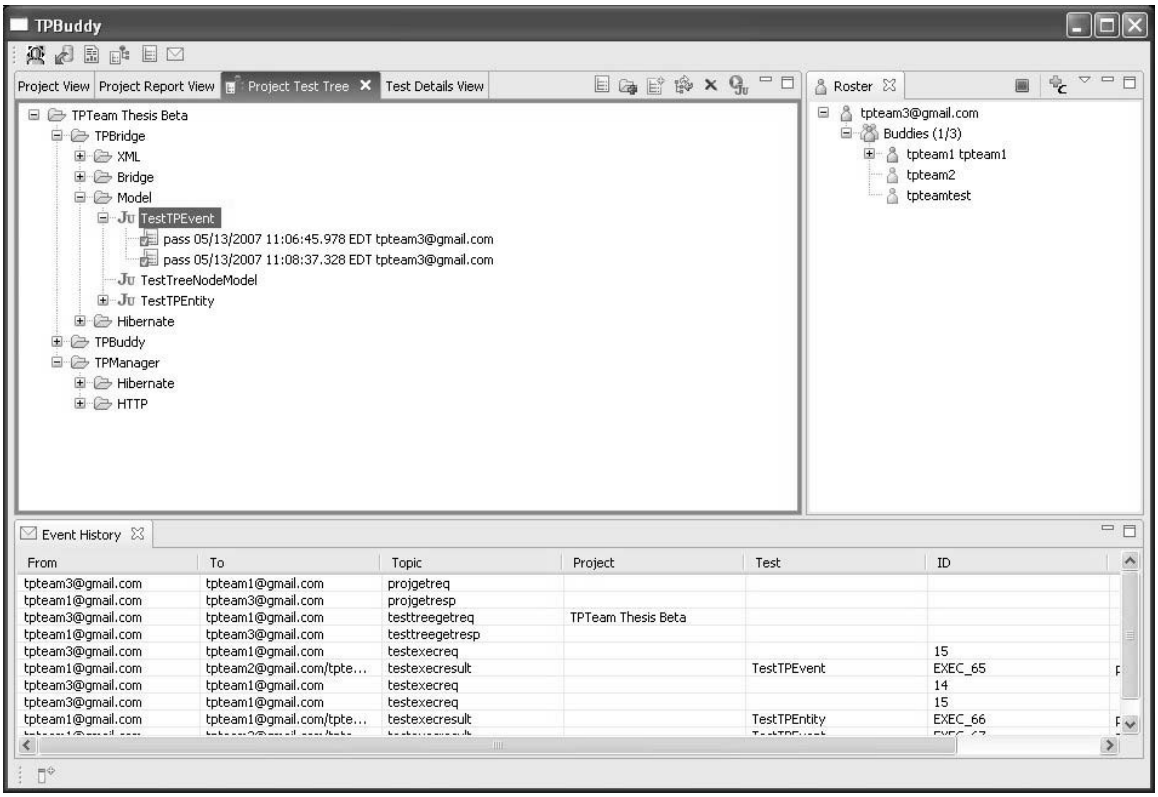

Figure 3 TPBuddy GUI in Action

Table 1 below summarizes the features of the TPTeam system. TPTeam exposes its features to users through the TPManager and TPBuddy components. The TPBridge is only used internally in a programmatic manner.

The TPManager is accessed by Web users through a browser or programmatically by TPBuddy RCP application clients. Web access to TPManager is broken down by two authorization roles: administrative and user. An administrative user can perform CRUD operations on all TPTeam objects, whereas the user role is more restricted. The TPBuddy features are similar to those provided for the Web user role.

Component	Description	Features
TPManager	An Eclipse runtime embedded within a Web application. Exposes functionality through a Web interface and XMPP messaging to clients.	For Web Administrative Role: • Product CRUD • Project CRUD • User CRUD • Test CRUD • Test Execution For Web User Role: • Product View • Project View • Account Update, View • Test CRUD • Test Execution
TPBuddy	An Eclipse RCP application, acts as a client to TPManager and peer to other TPBuddy instances.	• Project Details View • Project Graphical Reports • Test CRUD • Test Execution • TPBuddy Peer Instant Messaging

Table 1 TPTeam System Features

The remainder of this document will describe TPTeam in detail. Chapter 2 summarizes the various programming platforms and frameworks that were used to develop TPTeam. Next, Chapter 3 explains the object and relational models used in the design. This is followed by a user guide in Chapter 4 that describes how to install, configure, and run TPTeam. Finally, Chapter 5 gives the lessons learned and summarizes the results of the thesis project. An appendix listing the source code and configuration files is also included.

Chapter 2 Platforms and Frameworks Used

TPTeam builds upon and extends several open source software platforms and frameworks. The Eclipse open development platform provided many frameworks for TPTeam: Equinox, ECF, RCP, and TPTP. The object-relational mapping and database persistence for TPTeam was implemented with Hibernate, a Red Hat-sponsored open source project. TPTeam's event messaging system utilizes an Object-to-XML serialization library called Betwixt from the Apache Software Foundation (The Apache Software Foundation, 2007). Finally, the graphic reports that TPTeam displays were developed by using JFreeChart, an open source Java chart library. This chapter explains in detail the different platforms and frameworks used by TPTeam.

Eclipse

Eclipse aims to be a universal platform for the integration and creation of software tools and applications (The Eclipse Foundation, 2007c). It is famous as the leading Java integrated development environment (IDE). However, many people fail to realize that it also provides a large and active ecosystem of frameworks that can be used to build arbitrary applications. Eclipse supports a wide range of projects including but not limited to: IDE tools for different languages (Java, C, C++, Python), business reporting (The Eclipse Foundation, 2007d), Web application development (The Eclipse Foundation, 2007e), and application lifecycle management (The Eclipse Foundation, 2007f).

Much of the widespread adoption of the Eclipse platform and its diversity is owed to its architecture consisting of plug-ins and their run-time management system. The software building blocks of Eclipse are plug-ins. They are the smallest components that can be developed as separate entities and deployed into the Eclipse architecture.

The term "plug-in" is derived from the notion of extension points. An extension point is a publicly exposed Java interface. A plug-in may expose an extension point or elect to contribute to one. For example, if a plug-in X provides an extension point interface I, then a plug-in Y contributes to the extension point by providing a concrete implementation C of I. Plug-in X would then instantiate and run C on Y's behalf when Y is activated at run time.

Most of the Eclipse frameworks and its Java IDE itself consist of a cooperating set of plug-ins. Figure 4 below shows a simple representation of the set of plug-ins that make up the Eclipse Java IDE. Here the core plugs-ins within the Eclipse IDE can be seen as well as third-party plug-ins taking advantage of extension points. The Eclipse runtime is an implementation of the Open Services Gateway Initiative (OSGi, 2007a) specification and manages the lifecycle of the plug-ins. This runtime is known as the Equinox project within Eclipse. The Help, Team, and Workspace plug-ins are standalone components that provide a help system browser, CVS repository access, and Java project resource manipulations, respectively. The Workbench plug-in depends upon both the Standard Widget Toolkit (SWT) and JFace plug-ins. SWT provides access to the native windowing system widgets while JFace provides more sophisticated GUI components such as tree views and wizards built from SWT.

All of the TPTeam components are deployed as plug-ins, similar to the core and third-party one shown in Figure 4. The TPTeam plug-ins were developed using Eclipse 3.2.2 and Java 1.6 Standard Development Kit.

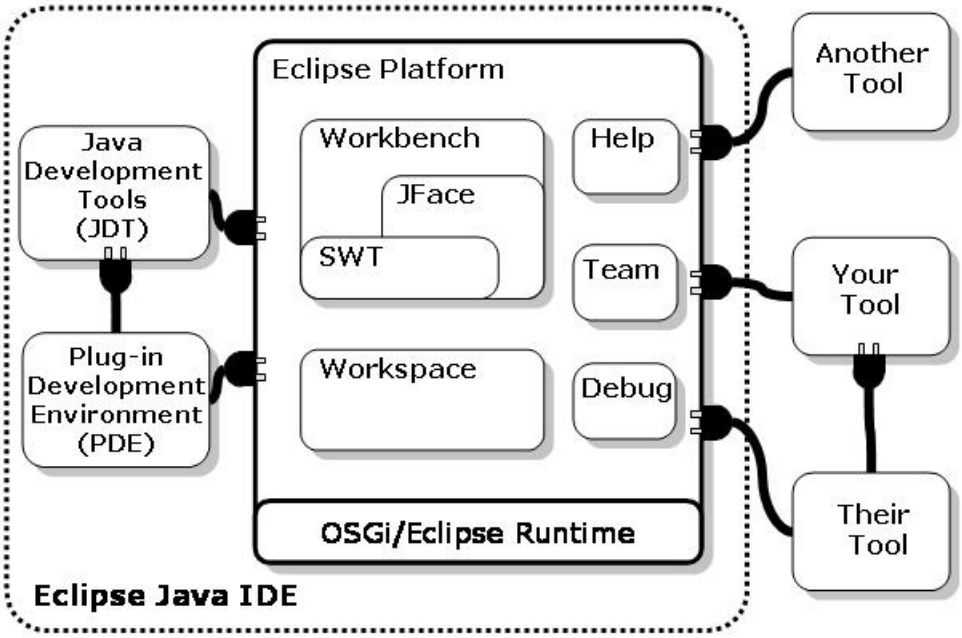

Figure 4 Eclipse Java IDE Architecture

The Service-Oriented Architecture (SOA) feature of the Equinox runtime plays a key role in TPTeam's messaging system. Equinox and the other Eclipse components used will be discussed in the following sections of this chapter.

Equinox

Equinox is the Eclipse implementation of the OSGi specification (The Eclipse Foundation, 2007g). It is a collection of bundles that also includes non-standard implementations that extend the framework, thought essential by Eclipse for the proper running and management of OSGi-based systems. OSGi is an independent, non-profit corporation whose goal is to define and promote an open specification for the deployment

of service applications on networked devices. Originally intended for embedded devices (similar to Java), it has been widely accepted by a wide variety of software vendors. The OSGi specification API is a Java framework. It was designed to solve problems such as: management of the software life-cycle, increased software portability by allowing applications to run unmodified in a heterogeneous environment, and the discovery/execution of other application services co-deployed on a device. OSGi implementations run within a JVM.

The OSGi framework has two major parts: a services platform and a deployment infrastructure. The services platform utilizes SOA to allow application bundles to find, bind to, and execute each other. The deployment infrastructure is used to install, start (activate), stop (deactivate), refresh, update, and uninstall application bundles. In OSGi, software components are physically packaged as jar files and referred to as bundles.

SOA is an OASIS reference model (OASIS, 2007b) that defines the use of loosely-coupled, interoperable application services. Interoperability is achieved by integrating services at the interface versus implementation level. The interface is typically specified through the use of a standardized API, such as that given by OSGi (OSGi, 2006b).

The OSGi form of SOA consists of three entities and three operations between them, as shown in Figure 5 below. The entities are:

- Service Provider Bundle: Allows Service Consumers to bind to and execute its published application services. Also publishes the availability of its services to the Service Registry. Registration includes the submittal

of service properties, name-value pairs that allow the Service Registry to differentiate service interfaces.

- Service Registry Bundle: Allows Service Requestors to query it for available services and the URI location of their respective Providers. Allows Service Providers to publish the availability of their services.
- Service Requestor Bundle: Finds Service Providers through the Service Registry. Binds to Service Providers in order to exchanges message and execute bundle services. Requestors may query the Service Registry with queries formatted in LDAP syntax.

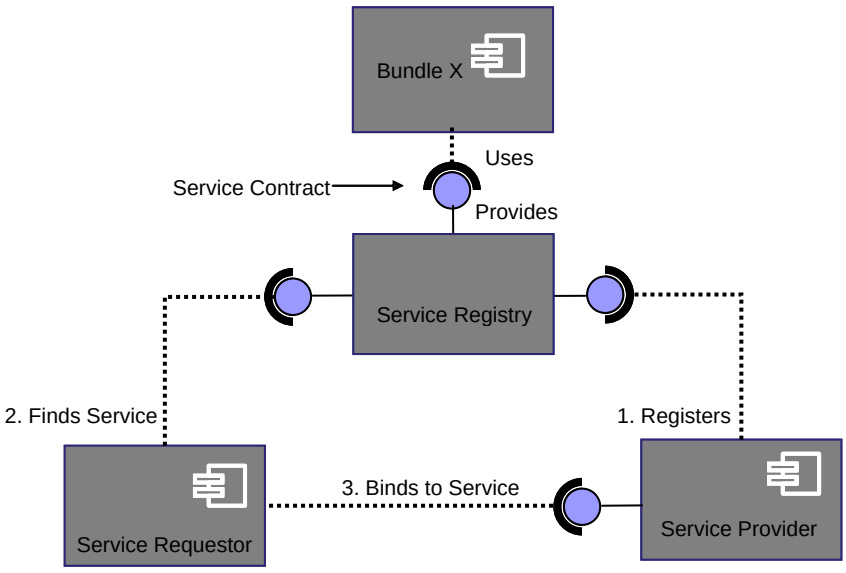

Figure 5 OSGi Service-Oriented Architecture

The platform interoperability and loosely-coupled characteristics of SOA are advantageous to OSGi. For example, the SOA interface encapsulates much of the technical implementation details and typically has a small footprint of between 10 to 200 Kbytes (Graham, Davis, & Simeonov, 2005).

OSGi also details the specifications for the Event Admin Service. This service provides for publish/subscribe event messaging, both in synchronous and asynchronous modes. The Event Admin Service represents a useful inter-bundle communication mechanism. See Figure 6 below for a class diagram of the Event Admin Service. One item of note is the lack of a standardized vocabulary or "topical lexicon" for the Event objects that the Event Admin Service handles. While the OSGi specification (OSGi, 2006c) does specify the details of the Bundle Events (concerned with lifecycle changes in bundles) and the Framework Events (for reporting about the lifecycle of the framework itself), the generic Event has more flexibility. The Event need only refer to an arbitrary String topic (e.g., "testexecreq" for a test execution request) and a collection of name-value properties.

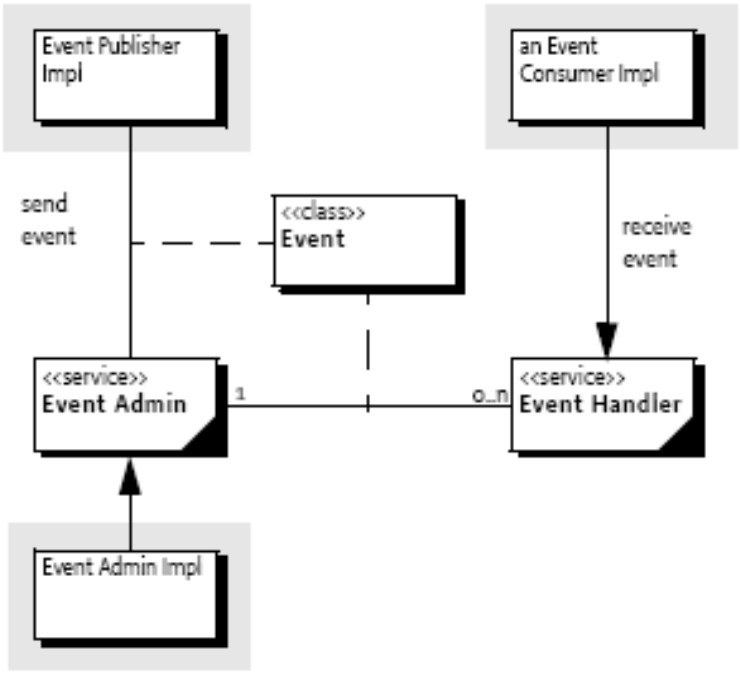

Figure 6 OSGi Event Admin Service Class Diagram (OSGi, 2006c)

The SOA and the Event Admin Service of Equinox are used extensively throughout TPTeam. The TPBridge, TPBuddy, and TPManager plug-ins each implement an eventadmin package. These packages provide an EventAdminClient class which publishes TPTeam events to the Event Admin service. An EventAdminHandler class is also implemented to subscribe to TPTeam events such as test execution result messages. The TPBridge plug-in exposes a service whereby clients can send ECF messages out-of-JVM. The design details of these classes are given in Chapter 3.

The deployment infrastructure of OSGi concerns itself with managing the lifecycle of bundles. The physical representation of bundles is in the form of a Java archive of files consisting of classes, supporting binary or text resources, and a manifest. The manifest file contains bundle metadata, such as dependencies on other bundles and which Java packages may be used by others. Eclipse plug-ins are configured as OSGi

bundles. This configuration is done automatically when a plug-in project is created with the Eclipse IDE. Thus, TPTeam plug-ins can take advantage of the OSGi lifecycle management.

Table 2 below summarizes the Equinox plug-ins used by this thesis project.

Equinox Plug-ins Used
org.eclipse.osgi_3.2.2.R32x_v20070118
org.eclipse.equinox.common_3.2.0.v20060603.
org.eclipse.equinox.registry_3.2.1.R32x_v20060814
org.eclipse.equinox.preferences_3.2.1.R32x_v20060717
org.eclipse.osgi.services_3.1.100.v20060601
org.eclipse.equinox.http_1.0.2.R32x_v20061218.
org.eclipse.equinox.event_1.0.0.v20060601a
org.eclipse.osgi.util_3.1.100.v20060601
org.eclipse.equinox.servlet.bridge.http_3.2.0.v20060510
org.eclipse.equinox.servlet.bridge.launcher_3.2.0.v20060510
org.eclipse.equinox.http.registry_3.2.0.v20060510

Table 2 Equinox Plug-ins Used by TPTeam

ECF

ECF is a framework of OSGi bundles and Java classes that provides an API for multi-protocol communication on the Eclipse platform. It achieves interoperability through protocol abstraction. The concept of a communication container and the Adapter design pattern are notions central to the understanding and use of ECF. While

the majority of ECF concerns itself with developing client-side applications, it does provide a generic protocol server that is simple to deploy and that others may build upon.

ECF communication containers are instances of the IContainer interface. An IContainer can represent point-to-point (client/server) as well as publish/subscribe (group) communications. Each IContainer exposes its functionality through its associated adapters. The adapters wrap the ECF API around the specific semantics of their protocol adaptee classes. A typical example of how an ECF IContainer would be used during its lifecycle is given in the code snippet below:

```
// Create shared object container for XMPPS provider
ISharedObjectContainer container =
SharedObjectContainerFactory.getDefault().
createSharedObjectContainer("ecf.xmpps.smack");


// Create ECF IDs from String IDs
ID targetID =
IDFactory.getDefault().createStringID("tpteam@gmail.com");

ID sharedObjectID =
IDFactory.getDefault().createStringID(TPSharedObject.class.getName());


// Create TPTeam shared object
TPSharedObject tpSharedObject = new TPSharedObject();


// Add shared object to container
container.getSharedObjectManager().addSharedObject(sharedObjectID,
tpSharedObject, new HashMap());


// Connect to XMPPS provider
container.connect(targetID, getJoinContext("password"));


// Broadcast TPEvent message to all peers out-of-JVM
tpSharedObject.getContext().sendMessage(null, tpEvent);


// Handle incoming messages within TPSharedObject object
public void handleEvent(Event event) {
 if (event instanceof ISharedObjectMessageEvent) {
ISharedObjectMessageEvent evt = (ISharedObjectMessageEvent) event
If(evt.getData() instanceof TPEvent)
```

```
{
      // Do stuff…
}


// Disconnect from XMPPS
container.dispose();
```

The ECF adapter types available and their supported protocols are summarized below in Table 3.

ECF Adapter Type	Adapter Description	Protocols Supported
IPresenceContainer	Allows registration of listeners for presence messages, text messages, and subscription requests. Provides interfaces for message sending, presence updates, and account management.	XMPP
ISharedObjectContainer	Manages the creation, access, and disposal of ISharedObjects	XMPP, JMS, ECF Generic Client/Server, ECF File Sharing Client, ECF Data Sharing Client
IChannelContainer	Exposes communication channel for asynchronous transport of byte arrays.	ECF Data Sharing Client
IFileShareContainer	Allows providers to expose file sharing to clients	ECF File Sharing Client
IDiscoveryContainer	Supports lookup to other Containers that support discovery.	ZeroConf/Bonjour

Table 3 ECF IContainer Adapter Summary

A few observations about ECF relative to the proposed work are worth noting. First, while ECF is implemented as a set of OSGi bundles, there is no existing "bridge" bundle that can integrate it with the Equinox Event Admin service. A bridge bundle would allow ALM events such as "execute test" to be routed through to the appropriate ECF Container and on to a remotely collaborating group of users or internal bundles. A bridging bundle would be highly desirable for such a use case. Secondly, ECF has provided an implementation of the XMPP protocol (XMPP Standards Foundation, 2007) that can be used with several well-established free providers such as Google Talk or Jabber.

TPTeam addresses both of the above points. It provides a TPBridge bridging bundle that allows both TPTeam plug-ins and arbitrary third-party bundles to subscribe to its Equinox SOA services. It also utilizes Google Talk as its XMPP communications provider. The ECF plug-ins used are given in Table 4 below. A detailed discussion of the TPBridge plug-in is given in Chapter 3.

ECF Plug-ins Used
org.eclipse.ecf.call_1.0.0.v20070330-2111
org.eclipse.ecf.discovery_1.0.0.v20070330-2111
org.eclipse.ecf.filetransfer_1.0.0.v20070330-2111
org.eclipse.ecf.identity_1.0.0.v20070330-2111
org.eclipse.ecf.presence.ui_1.0.0.v20070330-2111
org.eclipse.ecf.presence_1.0.0.v20070330-2111
org.eclipse.ecf.provider.xmpp_1.0.0.v20070330-2111
org.eclipse.ecf.provider_1.0.0.v20070330-2111
org.eclipse.ecf.sharedobject_1.0.0.v20070330-2111
org.eclipse.ecf.ui_1.0.0.v20070330-2111
org.eclipse.ecf_1.0.0.v20070330-2111
org.jivesoftware.smack_2.2.1

Table 4 ECF Plug-ins Used by TPTeam

TPTP

TPTP aims to be a universal platform for test and performance tools, (The Eclipse Foundation, 2007h). It was created as a top-level Eclipse project in 2004 through a donation of source code from IBM. TPTP itself is divided into four subprojects:

- Platform – Provides a GUI framework for common views, editors, and wizards used by other TPTP components. Defines a standard data model for information collection of test, trace, and log events. Supplies a common Agent Controller execution framework for the deployment, launch, and management of TPTP tests and applications.

- Test - Extends the platform to provide value-added TPTP versions of JUnit and JUnit plug-in tests. Makes available an automation client that can deploy, launch, and interrogate TPTP tests from outside the Eclipse environment.
- Tracing and Profiling - Extends the platform to provide tracing and profiling tools. Has a JVM monitoring agent that collects and analyzes heap and stack performance during runtime.
- Monitoring Tools – Extends the platform by providing tools for monitoring application server and system performance.

TPTeam actively utilizes only the Test project of the TPTP platform. In particular, it takes advantage of the TPTP JUnit and JUnit Plug-in tests ability to be launched and interrogated by the automation services client that the TPTP tools project provides. TPTeam's TPManager plug-in uses the automation client to communicate with the Agent Controller in order to run test executions on it's behalf on the test target machine. Figure 7 below shows the TPTP components utilized by TPManager and the target machine under test.

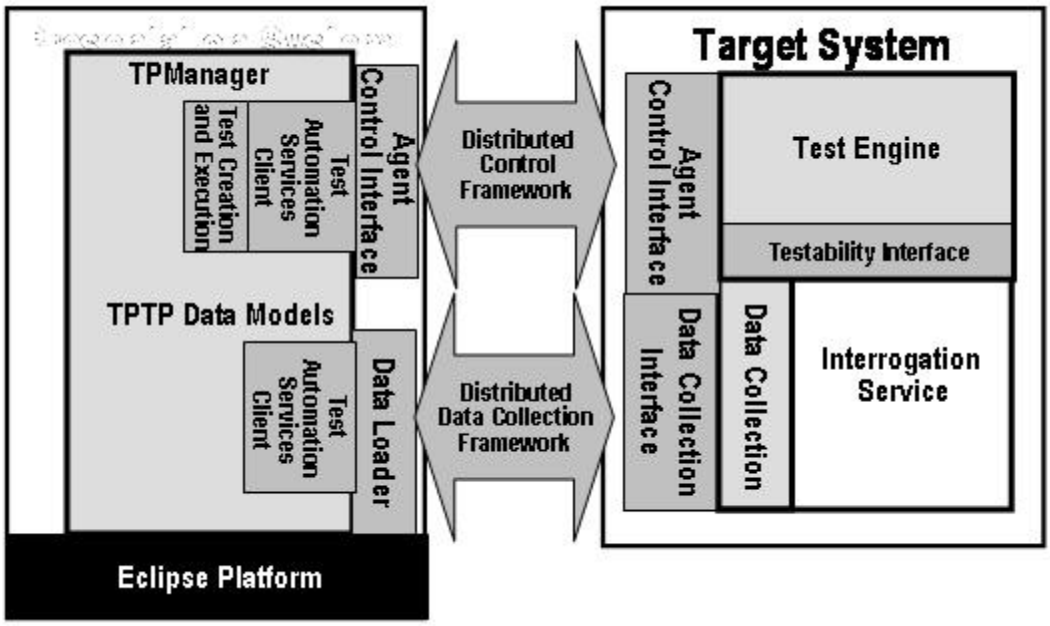

Figure 7 TPManager Use of the TPTP Automation Client

The automation services client is packaged within a jar file that the TPTP test tools project provides. TPManager instantiates an instance of the TPTP AutomationClientAdapter class for each test execution request it receives. Next, it gives the AutomationClientAdapter instance the necessary properties in order to connect with the target system's Agent Controller: URL connection string, eclipse home directory, test workspace directory, test project within the workspace, and finally the TPTP testsuites within the project to be executed. Finally, TPManager's automation client can interrogate the TPTP framework to get the overall verdict of each test suite execution.

In order to take advantage of the framework shown in Figure 8 above, one needs to package his or her tests as TPTP JUnit tests. Doing so is a straightforward procedure and TPTP provides several wizard GUIs for this purpose. For example, Figure 8 below shows an example of the TPTP JUnit test wizard. It is similar to the JUnit wizard that the

Eclipse Java IDE provides, with the exception that it provides additional tabbed pages for creating new test methods and assigning invocations to those methods (e.g., a loop of invocations or single executions). TPTP JUnit testsuites are stored in a binary format that conforms to the platform's standard data model. The extra effort of creating JUnit tests in TPTP format allows for the headless automated test execution scenario of Figure 7.

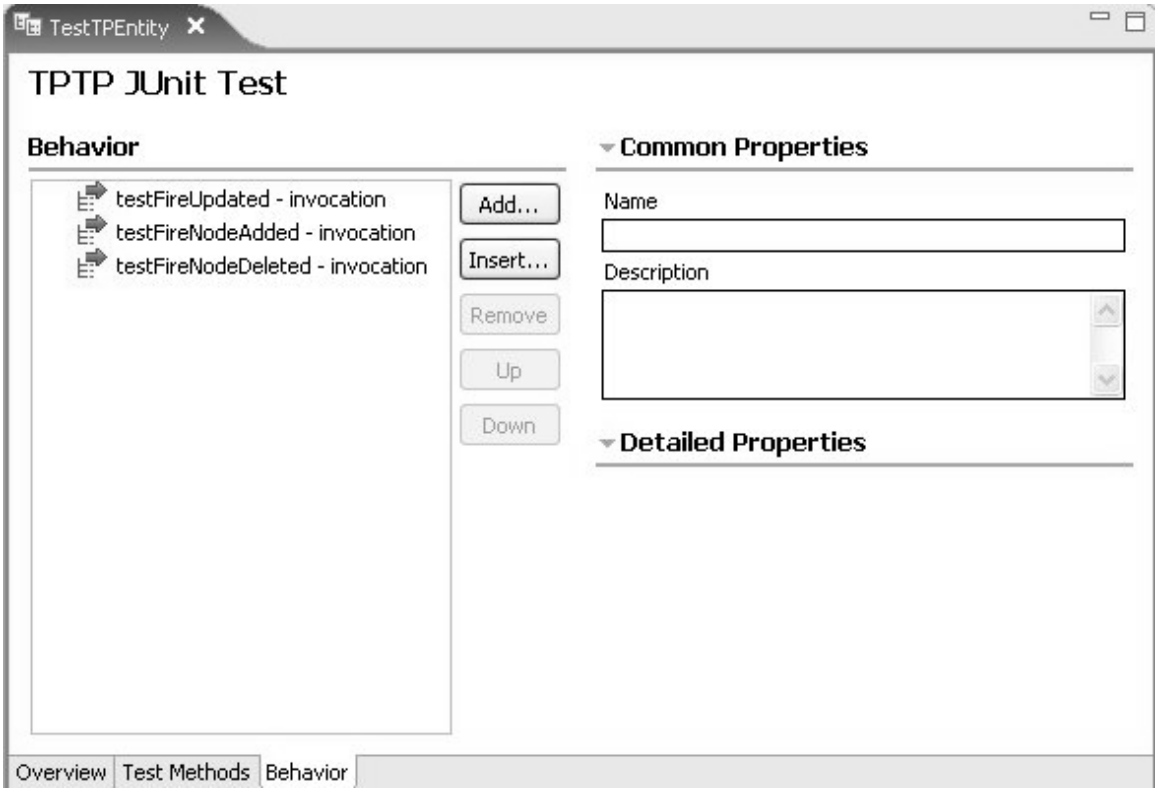

Figure 8 TPTP JUnit Tabbed Wizard Pages

Table 5 below summarizes The TPTP components used in this thesis project.

TPTP Component Used
TPTP-4.2.2 Testing Tools Runtime Plug-in Set
Agent Controller TPTP-4.2.2 Windows IA32 Distribution

Table 5 TPTP Components Used by TPTeam

RCP

Eclipse RCP applications (The Eclipse Foundation, 2007i) take advantage of the robust plug-in component model and Equinox run-time management system mentioned previously in this chapter. This arrangement makes the deployment and maintenance of client software easier for IT managers. For example, the banking giant JP Morgan decided to convert their existing financial modeling suite of Excel spreadsheets and custom C++ code to an Eclipse RCP application (The Eclipse Foundation, 2007j). The ability to share a common set of plug-in components with the option to easily extend or add other plug-ins on an ad hoc basis was cited as a reason for the switch. JP Morgan also reported use of native windowing widgets, high-speed local computation, OS/machine portability, and time to release as advantages obtained by using Eclipse RCP

The TPBuddy RCP application of TPTeam seeks the same benefits as realized by JP Morgan. Implementing the TPBuddy plug-in as an addition to the Eclipse Java IDE would require the overhead of more than 300 plug-ins with an IBM Callisto "convenience" Release. Packaging TPBuddy as an RCP application results in a svelte and easy to manage set of about thirty-five plug-ins.

The TPTeam RCP application consists of the TPBuddy and TPBridge plug-ins resting on top of the RCP plug-in stack. Beneath the TPBridge component are the ECF

set of plug-ins needed for real-time collaborative communication via serialized object and instant messaging. The RCP plug-ins used in the development and deployment of TPTeam were from the Eclipse 3.2.2 RCP SDK, as shown in Table 6. This application stack is shown below in Figure 9.

RCP Component Used
eclipse-RCP-SDK-3.2.2-win32 Plug-in Set

Table 6 RCP Components Used by TPTeam

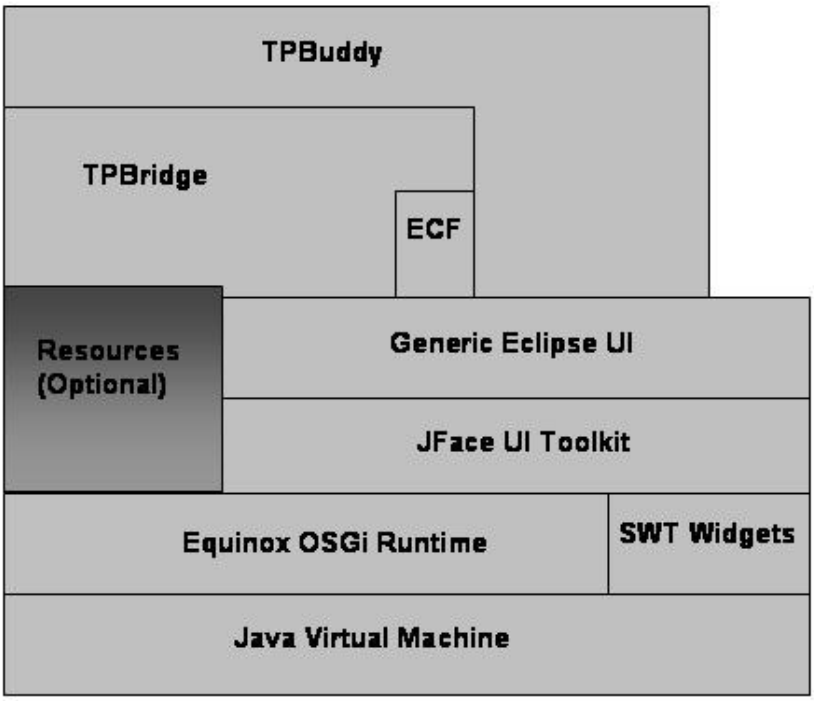

Figure 9 TPTeam RCP Application Stack

Hibernate

It has been estimated that 30% of all Java application code written has been dedicated to bridging the gap between object-oriented business logic and relational database systems (Bauer and King, 2005). There is an "impedance mismatch" between the object and relational models that makes software development in this case complex, cumbersome, and costly.

Hibernate is an object/relational mapping (ORM) framework designed to overcome the previously mentioned impedance (Red Hat, 20007). It utilizes plaintext metadata mapping files that describe the ORM from an application's Java objects to the tabular representation given in the database. Once the Hibernate ORM metadata files are configured, a developer can code with the framework's intuitive object-oriented API. The Hibernate middleware libraries then perform the necessary bridging to convert the object CRUD operations to the relational model.

TPTeam utilized Hibernate to:

- Reverse engineer a normalized form of the TPTeam relational database model into a working object model.
- Replace low-level Java Database Connectivity (JDBC) API calls in SQL with more concise and intuitive object-oriented Hibernate Query Language (HQL) CRUD statements.
- Forward engineer from Hibernate mapping files to the Data Definition Language (DDL) statements needed to create a relational database.

Table 7 below shows the Hibernate components used by TPTeam.

Hibernate Component Used
Hibernate Core 3.2.2 (zip file of jars)
Hibernate Tools 3.2b9 (zip file of plug-ins)

Table 7 Hibernate Components Used by TPTeam

Betwixt

Betwixt is a library for transforming JavaBeans (Sun Micro Systems, 2007b) into XML and *vice versa*. It offers more customizable features than the standard object serialization offered by the Java 1.6 standard development kit (SDK). For example, the SDK object serializer forces one to recursively serialize all object properties of a JavaBean unless labeled with the transient or static modifier. If the JavaBeans to be serialized is also part of a persistence model such as Hibernate, then no property stored in the database can use a transient modifier. Thus, the result of applying SDK object serialization to a Hibernate domain object can lead to unnecessarily large output data when only top-level information is desired.

Betwixt, on the other hand, can be used to solve the unwanted recursive transformation of Hibernate JavaBeans. Its design utilizes a .betwixt file to override the default introspection done by the SDK java.bean.Introspector class. This customization is done without altering the JavaBeans and hence the persistence behavior of Hibernate. A sample .betwixt file for the TPTeam Test class is shown below.

```
<?xml version='1.0' encoding='UTF-8' ?>
<info primitiveTypes="attribute">
      <hide property="testExecutions" />
      <element name='test'>
            <addDefaults/>
      </element>
</info>
```

The above .betwixt file instructs Betwixt to insert all primitive Java types as attributes within the top-level element tag of the XML serialization. Furthermore, it will override the default mapping and not include any test execution information associated with a given test. The number of test executions associated with a given test could exceed tens of thousands. Finally, the top-level tag of the serialization will have the name “test” with all other default XML mappings intact. The method excerpt below shows how simple it is to use the Betwixt BeanWriter class to map a List of TPTeam Project objects to XML.

```
public static String getXML(Set<Project> projs)
{
      ByteArrayOutputStream baos = null;
      try {
            baos = new ByteArrayOutputStream();
            BeanWriter bWriter = new BeanWriter(baos);
            bWriter.setWriteEmptyElements(false);
            bWriter.enablePrettyPrint();
            bWriter.writeXmlDeclaration("<?xml version='1.0' ?>");
            bWriter.writeXmlDeclaration("<projects>");
            for(Project proj : projs)
                  bWriter.write("project", proj);
                  bWriter.writeXmlDeclaration("</projects>");
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
      }
```

TPTeam used Betwixt in order to minimize the size of TPEvent messages sent across the Internet via Google Talk. A quantitative discussion of the reasoning behind

this choice is given in Chapter 5 of this report. Table 8 below summarizes the Betwixt libraries used in TPTeam.

Betwixt and Related Libraries Used
commons-betwixt 0.8
commons-beanutils-core 1.7
commons-digester 1.7
commons-logging 1.0.4
xerces 2.4
xml-apis 1.0.b2

Table 8 Betwixt and Related Library Usage in TPTeam

JFreeChart

JFreeChart is an open source Java charting library (Gilbert, 2007). It was selected over other Java charting frameworks because it has the following characteristics:

- Free and open source, released under the GNU license.
- Provides chart types that meet TPTeam requirements: Pie, Bar, and Timeline.
- Extremely short (one hour) time to productive output.
- Creates charts embedded in SWT widgets for display within Eclipse views.
- Small library footprint of a few megabytes.
- Well documented online and with a hardcopy 699 page developer's guide.

In fact, JFreeChart was the *only* software the author could find that met the first four of the above criteria. For example, the Eclipse Business Intelligence and Reporting Toolkit (BIRT, 2007) is a well-documented (Weathersby, French, Bondur, Tatchel, & Chatalbasheva, 2006; Peh, Hannemann, & Hague, 2006) and aggressively marketed open source graphical tool. However, after over twelve hours of frustrating deployment and debugging example projects downloaded from the Eclipse website, the author abandoned BIRT. With JFreeChart, on the other hand, the author had coded a working SWT pie chart view in TPBuddy after only one hour. BIRT components add tens of megabytes of binary file overhead to an application. JFreeChart requires an order of magnitude less footprint.

Table 9 below shows the JFreeChart components used by TPTeam

JFreeChart and Related Libraries Used
jfreechart-1.0.5
jfreechart-1.0.5-experimental
jfreechart-1.0.5-swt
jcommon-1.0.9
swtgraphics2d

Table 9 JFreeChart and Related Libraries Used by TPTeam

Now that the building blocks of TPTeam have been discussed, its design can be put into context and covered in detail. This will be done in the next chapter.

Chapter 3 Design

This chapter provides an overview of the TPTeam design. It begins with a system level discussion of the top-level components TPBridge, TPBuddy, and TPManager. Next, pertinent object model designs are reviewed for each of the three top-level components. Finally, the database relational model is explained. This chapter, combined with a close examination of the source code, should provide a software engineer with enough information to extend or modify TPTeam to suite his or her own purposes.

System Architecture

There are three main software entities in TPTeam: TPBridge, TPBuddy, and TPManager. The TPBridge plug-in combines with each of TPBuddy and TPManager to form two separately deployed run-time components: the TPBuddy/TPBridge RCP application and the TPManager/TPBridge embedded eclipse Web application.

Distributed serialized TPEvent objects are passed between all three TPTeam components. Each TPEvent object contains a String topic indicating whether it is a test execution request, test result, etc., plus a HashMap of TPTeam Meta data such as id, parent container, JUnit test execution verdict, and so forth. The TPTeam components utilize the Equinox built-in OSGi EventAdmin service to pass TPEvents in a publish/subscribe manner. The TPBridge was partly inspired by a published account of an OSGi-JMS Bridge (Donsez, 2007). Both bridges are event-driven through the OSGi

SOA infrastructure. Figure 10 below shows the event-driven interactions between the TPBuddy, EventAdmin, and TPBridge bundles which together form a standalone RCP application.

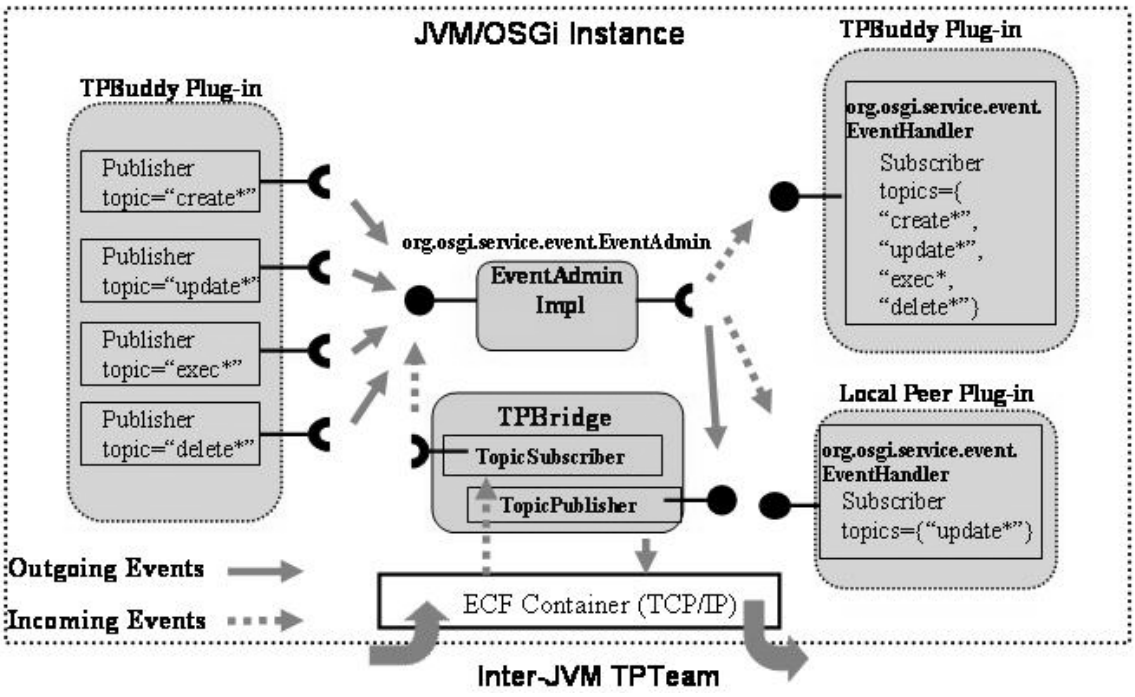

Figure 10 TPBuddy/TPBridge System Architecture

The local TPBuddy will need to notify remote collaborating TPTeam and local interested plug-ins of the TPEvents that it issues. This is done by having it get a reference to the EventAdmin Service from the Equinox Service Registry on bundle startup. Next, whenever it needs to perform a TPTeam operation, it will send a CRUD or execution TPEvent to the EventAdmin Service. This will be followed by the EventAdmin Service routing the message to all EventHandler Service implementers registered with the service who have subscribed to the TPTeam topics. Only local bundles will receive these messages. There is no built-in mechanism within the OSGi

specification or Equinox implementation to route messages outside a given JVM. The *ad-hoc* TPBridge is required for message routing beyond the local JVM.

The TPBridge bundle provides the required inter-JVM routing functionality. The TPBridge keeps a mapping of the TPSharedObjects that it contains. Each TPSharedObject broadcasts serialized messages over XMPP to all TPTeam peers that are subscribed to it. Each TPSharedObject of the TPBridge can be found by filtering on the TPSharedObject ID. The bridge parses messages from the EventAdmin Service, retrieves the corresponding TPSharedObject, and sends the TPEvent message to all inter-JVM TPTeam bundles subscribed to the TPSharedObject.

The local TPBuddy will receive incoming TPEvent messages from local and remote TPTeam bundles. These incoming events correspond to the case of an updated test project tree data structure or the results from a test execution. The local TPBridge will first receive a TPEvent from one of the TPSharedObjects within its collection. It will parse the message and issue an event to the Event Admin Service. Next, all local plug-ins that registered as EventHandler services for TPEvents will receive the TPEvent. The local TPBuddy will receive the message, parse it, and update its test project tree and other collaboration views in real-time.

The TPManager plug-in works in an identical fashion as TPBuddy. It sends and receives TPEvents via the EventAdmin service and its companion TPBridge in the same way as TPBuddy plug-ins. However, there is only one instance of a TPManager running per TPTeam installation. Furthermore, the singleton instance TPManager/TPBridge deployment is embedded within a servlet container inside an OSGi runtime. The TPManager plug-in handles TPEvent topics that are complimentary to those of the

TPBuddy. For example, a TPBuddy may issue events having a topic “testexecreq” representing a test execution request. The TPManager, on the other hand, would handle that request and then issue a corresponding TPEvent with topic “testexecresp” indicating a response to the execution request. The TPManager architecture is given in Figure 11 below.

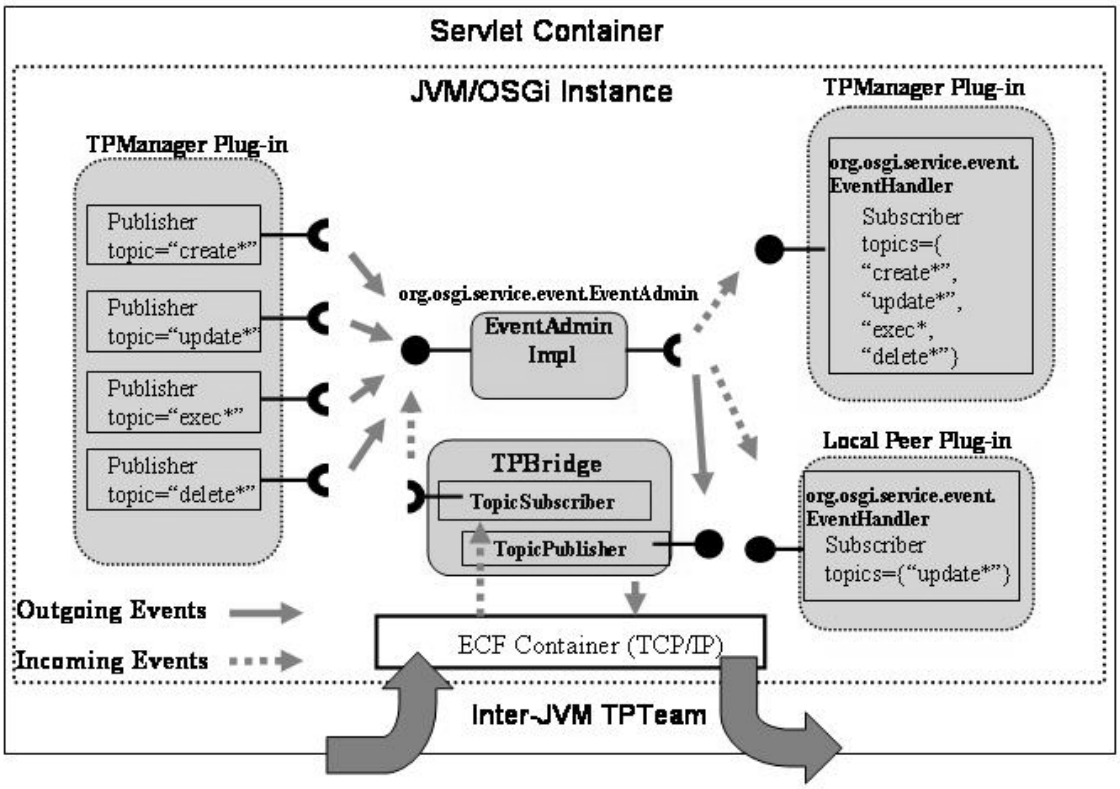

Figure 11 TPManager/TPBridge System Architecture, Part I

The TPManager exposes servlets through the OSGi HttpService service. Additionally, through the use of the Equinox ServletBridge package, it runs its OSGi runtime within a servlet container, such as Apache Tomcat. This configuration allows the TPManager to respond to Web browser client HTTP requests as well as those from TPBuddies over XMPP. Figure 12 below shows the architectural components of TPManager configured to service HTTP requests from clients.

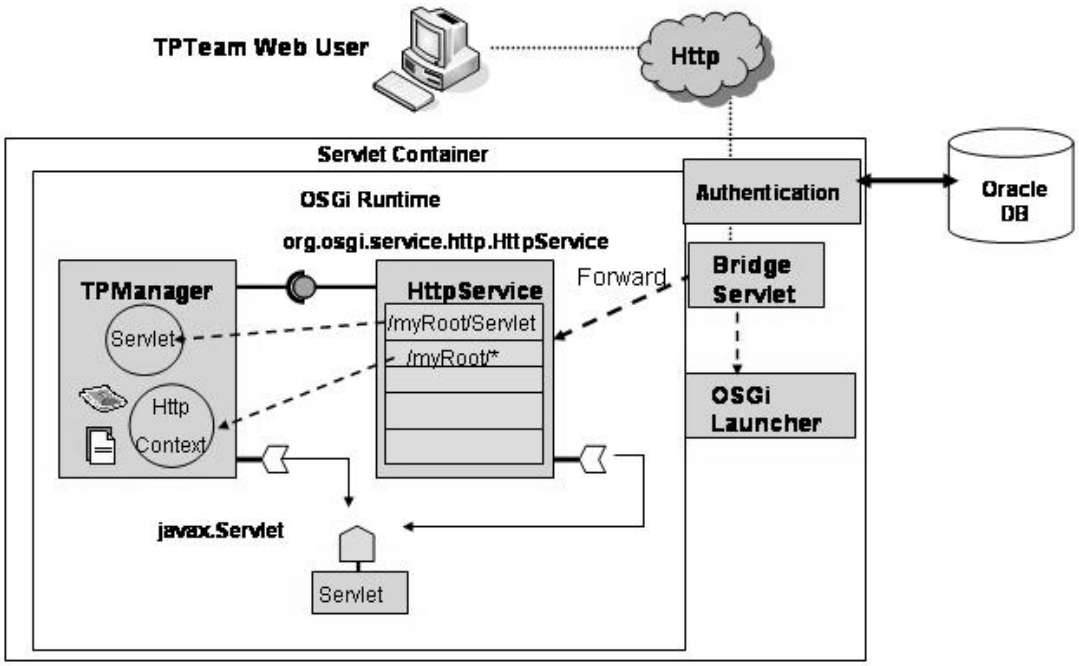

Figure 12 TPBridge System Architecture, Part II

The system takes advantage of the Tomcat servlet container's built-in JDBCRealm (Tomcat, 2007). The JDBCRealm allows one to specify in a server.xml configuration file the database parameters needed to authenticate a user against an Oracle database (shown as "Oracle DB" in Figure 12). Once the JDBCRealm properties have been specified, Tomcat intercepts authentication requests, performs authentication against the database, then either sets the necessary HTTP session variables for authorization or routes denied users to a login error page. The TPManager combined HTTP/ECF scenario is shown in Figure 13 below.

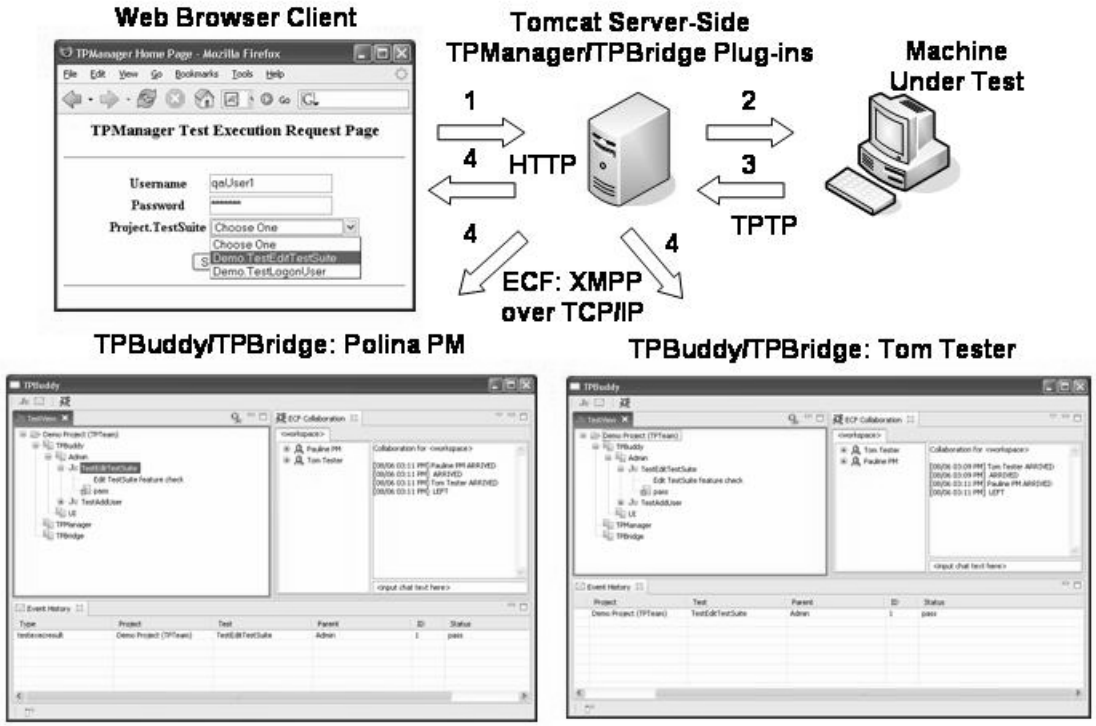

Figure 13 TPManager/TPBridge Combined HTTP/ECF Scenario

The following sections in this chapter will cover in detail the pertinent object models that were designed to support the above system architecture.

Object Model

The object model of the project includes the design of the TPBuddy, TPBridge, and TPManager plug-ins. The TPBridge and TPManager are non-GUI bundles. The TPBuddy and TPManager bundles depend upon the TPBridge classes. For this reason, The TPBridge object model will be discussed first.

TPBridge

The TPBridge plug-in consists of seven packages. They are listed in Table 10 below. The significant packages in terms of object model design are: bridge, eventadmin,

hibernate, and model. The class diagrams of these packages will be discussed in the following sections.

TPBridge Package	Functionality
NA (base)	Holds OSGi Activator class that controls the starting and stopping of the plug-in as well as initialization of provided OSGi services and event handlers.
bridge	Contains classes that provide the API contract for use of ECF communications.
chart	Provides helper and model classes for dealing with graphical displays of TPTeam state.
eventadmin	OSGi event handling classes.
hibernate	Hibernate framework classes for TPTeam.
model	Classes that model TPEvent entities.
xml	Utility classes for the serialization of TPTeam objects into XML.

Table 10 TPBridge Packages (edu.harvard.fas.rbrady.tpteam.tpbridge.* Prefix)

Bridge package class diagram. The bridge package is shown in Figure 14 below. This package exposes its functionality by providing an OSGi service to client plug-ins within a JVM.

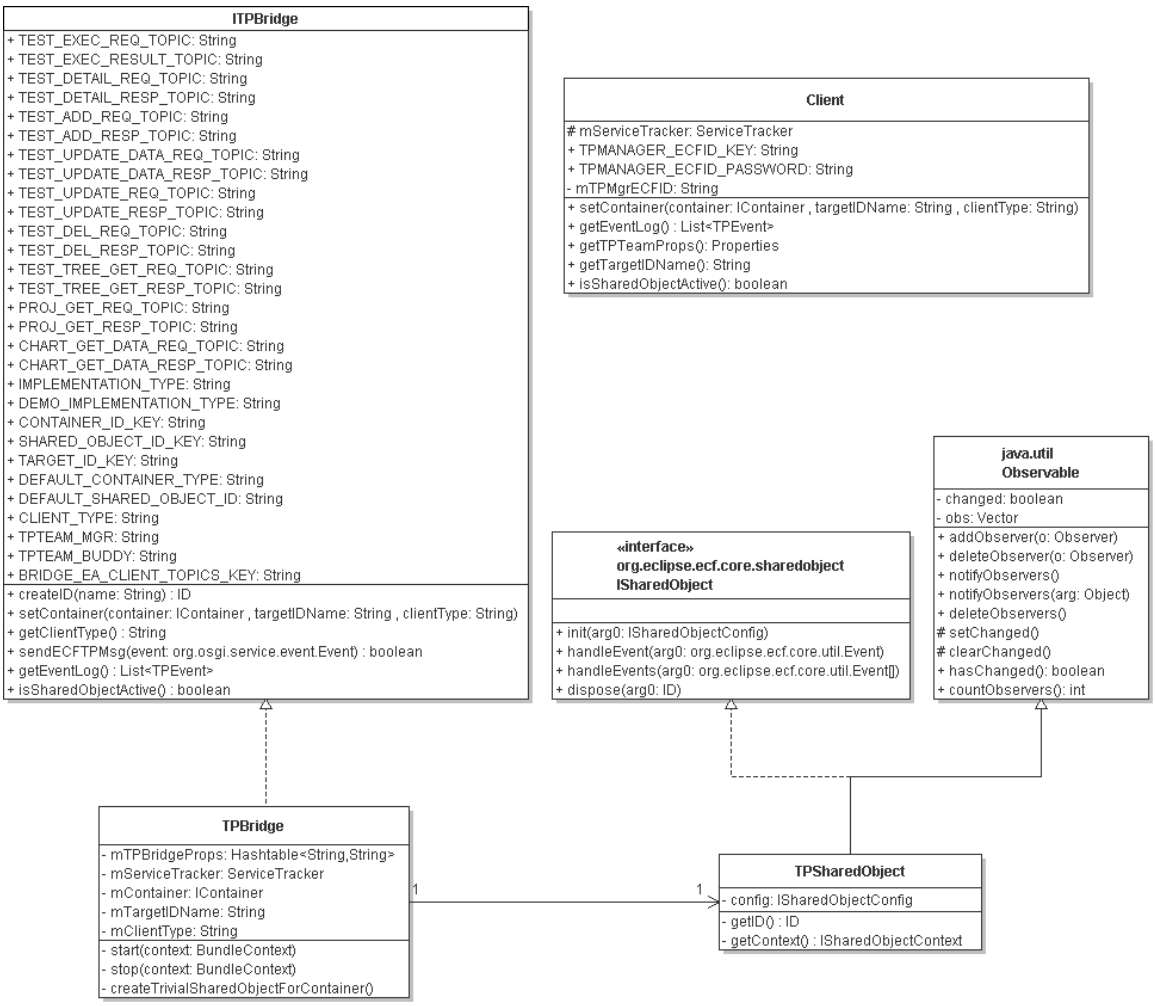

Figure 14 Package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge Class Diagram

It also contains a helper Client class that provides convenience methods that wrap much of the functionality of the bridge OSGi service. The ITPBridge interface defines the API contract that clients have with the package. The TPBridge class implements that interface and has a one-to-one relationship with a TPSharedObject . During the plug-in start phase, the TPBridge object will register itself as an OSGi service and then wait for requests. Once a request to set its ECF container is received, the TPBridge object will instantiate a singleton TPSharedObject and bind it to the given ECF container.

Inter-JVM messages are sent by invoking the bridge service's sendECFTPmsg method. This method will construct a TPEvent from the passed-in OSGi Event object parameter and then send it as a serialized object message via the TPSharedObject member.

The TPSharedObject class implements the ECF SharedObject interface as well as extends the standard Java class Observable. This inheritance allows a TPSharedObject to be notified if any messaging events from ECF peers have been received and also to notify any of its registered observers when that happens. This mechanism is particularly useful when an OSGi event handler object needs to be notified about an incoming ECF TPTeam event from outside the JVM. The eventadmin package, which will be discussed next, takes advantage of this mechanism.

Eventadmin package class diagram. The eventadmin package is shown in Figure 15 below. There are two classes: EventAdminHandler and EventAdminClient. The EventAdminHandler is used to handle OSGi and ECF events. The EventAdminClient is used to send OSGi events to the OSGi EventAdmin service.

The EventAdminHandler implements the OSGi EventHandler interface so that it may intercept and forward events to out-of-the JVM peers via the TPBridge object. It obtains a list of event topics that it should subscribe to from the plug-in Activator. Any OSGi events matching a topic on the list will be sent out-of-JVM by the TPBridge object.

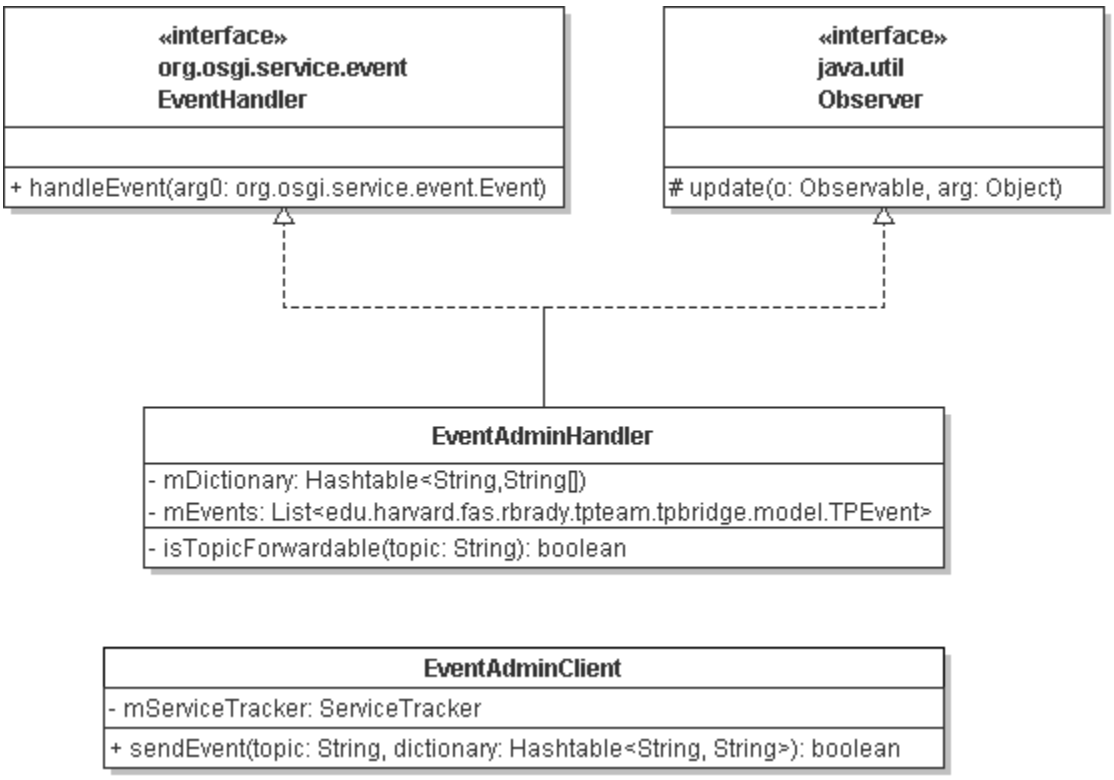

Figure 15 edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin Class Diagram

The EventAdminHandler implements the Observer interface in order to receive any out-of-JVM messages received by the singleton TPSharedObject. When updated with an ECF event, the EventAdminHandler will extract the embedded TPEvent and use the EventAdminClient client to forward the corresponding OSGi event to all subscribed plug-ins within the JVM. It holds a member variable of a List of the TPEvents that were received during a given session. The List may be used for tracking or log information by third-party plug-ins.

The EventAdminClient object sends OSGi events through the EventAdmin service API. It maintains a reference to the service from its ServiceTracker member variable, which it binds to during plug-in startup.

Hibernate package class diagram. The hibernate package is shown below in Figure 16. This package provides the basis from which the relational model is derived. Objects for the management of tests, users, and projects are present in the model.

The Test class is central to the model. It must be related to a TestType (e.g., "Folder" or "JUnit") object, has a composition of TestExecution objects, and can optionally map to a JunitTest object. Helper methods initSkeleton() and initProps() are provided with the Test class so that during XML serialization only the properities needed are lazily loaded. The TestExecution class implements a compareTo() method so that executions can be sorted by the date of the completion.

The Product, Project, TpteamUser, and Role classes make up the remainder of the model. The Project class is central to this portion. It must be related to a single Product (e.g., "TPTeam Version 0.1"), has a composition of Test objects, and a many-to-many mapping to TpteamUser. A helper method initSkeleton() is implemented to make XML serialization more efficient, for the same reasons as given for the Test class. The Role class is provided to allow for different levels of access to TPTeam via the HTTP interface.

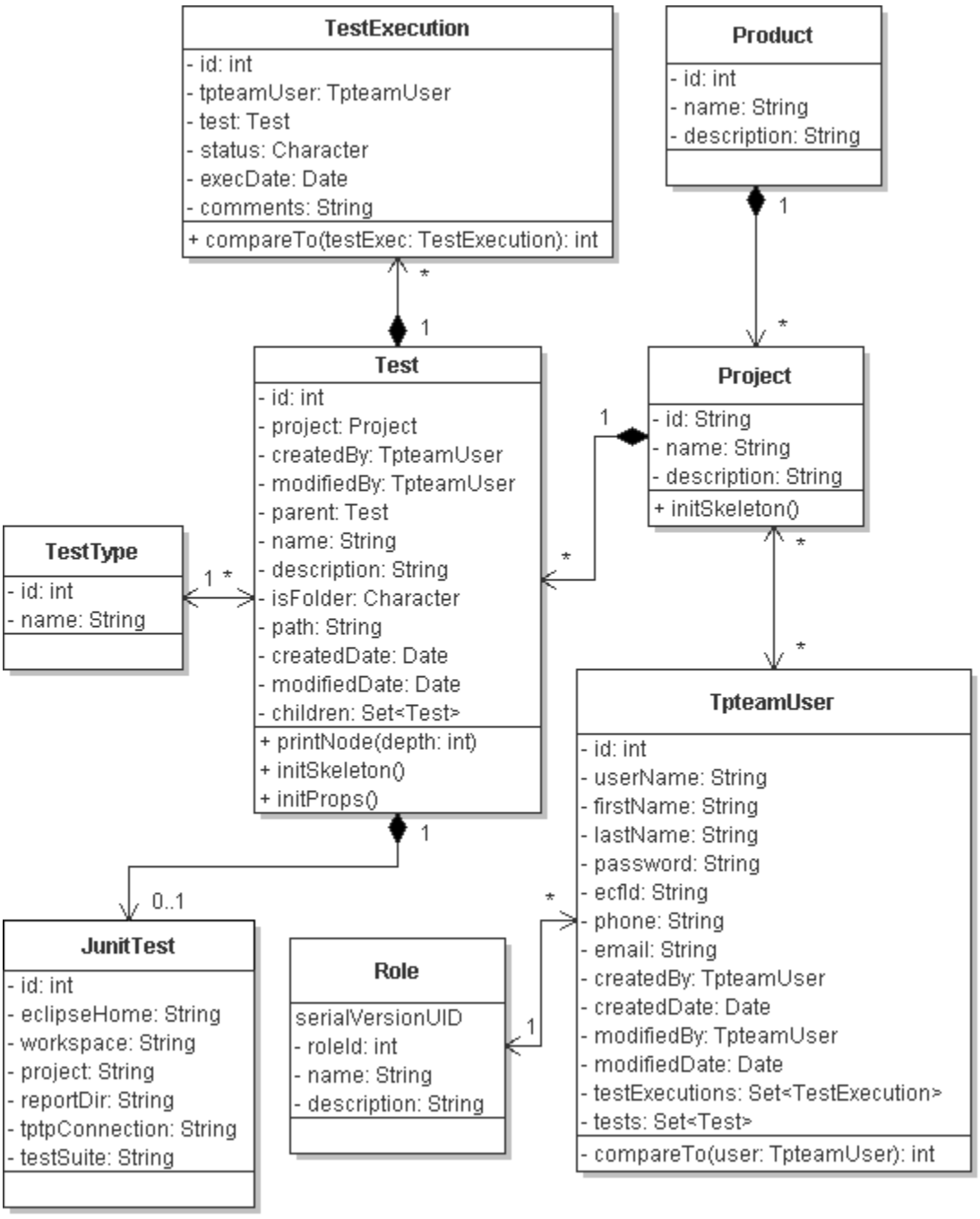

Figure 16 Package edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate Class Diagram

Model package class diagram. Figure 17 below shows the class diagram for the model package. This package is used to represent TPTeam entities as tree data structures and also for the use of TPEvents.

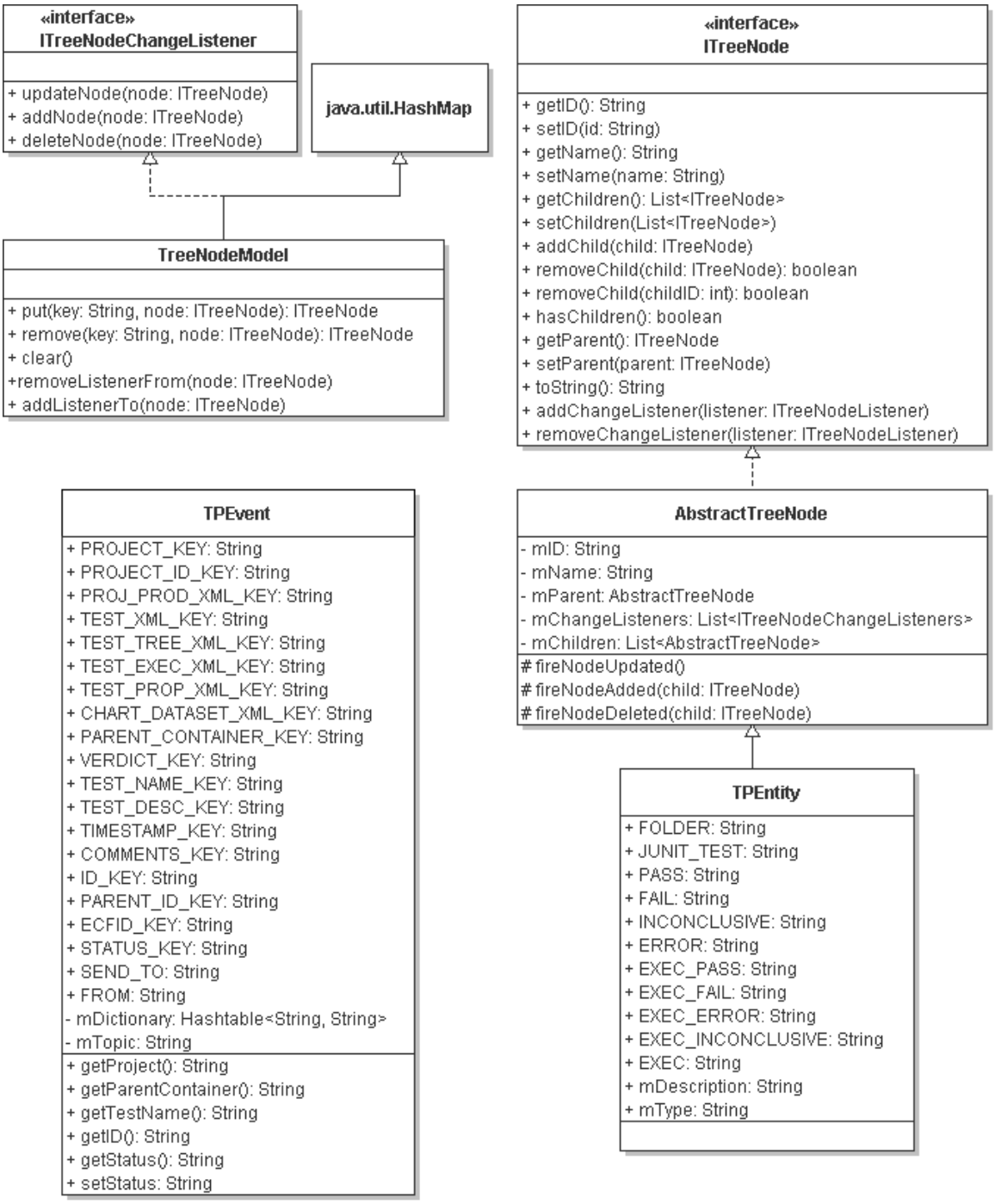

Figure 17 Package edu.harvard.fas.rbrady.tpteam.tpbridge.model Class Diagram

The TPEntity hierarchy of the package is used for serializing stubbed versions of Test and TestExecution objects and their direct loading into an Eclipse RCP tree view. TPEntity implements a composite design pattern by extending AbractTreeNode. It provides fireNode* events in order to maintain Model-View-Controller synchronization between the TPEntity model and any ITreeNodeChangeListeners that may be observing changes to the model.

The TreeNodeModel class was created to provide a data structure with O(1) read operations on the TPEntity tree it contains. This is important so that the Eclipse RCP view in TPBuddy can be quickly updated. It implements the ITreeNodeChangeListener interface so that it can keep its HashMap synchronized with the TPEntity model.

Finally, the model package also contains the TPEvent class. All messages within the top-level TPTeam components are sent as instances of this class. TPEvent acts as an adapter of OSGi Event objects as well as providing String constants and convenience methods for TPTeam property queries.

TPBuddy

The TPBridge plug-in consists of eight packages. They are listed in Table 11 below. The significant packages in terms of object model design are: charts, eventadmin, views. The class diagrams of these packages will be discussed in the following sections.

TPBuddy Package	Functionality
NA (base)	Holds OSGi Activator class that controls the starting and stopping of the plug-in as well as initialization of TPBridge client and OSGi event handler/client.
actions	Contains UI Action classes associated with UI icons and used for opening the various TPBuddy views.
charts	Provides helper and model classes for dealing with graphical displays of TPTeam state.
dialogs	Modal dialogs for adding and updating Tests objects.
eventadmin	OSGi event messaging classes.
tpbridge	Provides the client to the TPBridge exposed OSGi service.
views	Views and Controllers for all of TPBuddy.
wizard	Utility wizard for logging into TPTeam.

Table 11 TPBuddy Packages (edu.harvard.fas.rbrady.tpteam.tpbuddy.* Prefix)

Charts package class diagram. Figure 18 below shows the class diagram for the charts package. At the top of the hierarchy is the class AbstractChart. It provides final member variables that represent the different colors to be displayed in the TPBuddy charts, the String labels for test verdicts, and abstract methods for creating charts. Three classes extend AbstractChart: PieChart, BarChart, and LineChart. These classes are used for displaying pie, bar, and timeline charts, respectively.

Each of the PieChart, BarChart, and LineChart classes has similar behavior. They all take a TPTeam ChartDataSet array and through polymorphism at runtime convert it to the appropriate JFreeChart AbstractDataSet. Similarly, the appropriate chart is created

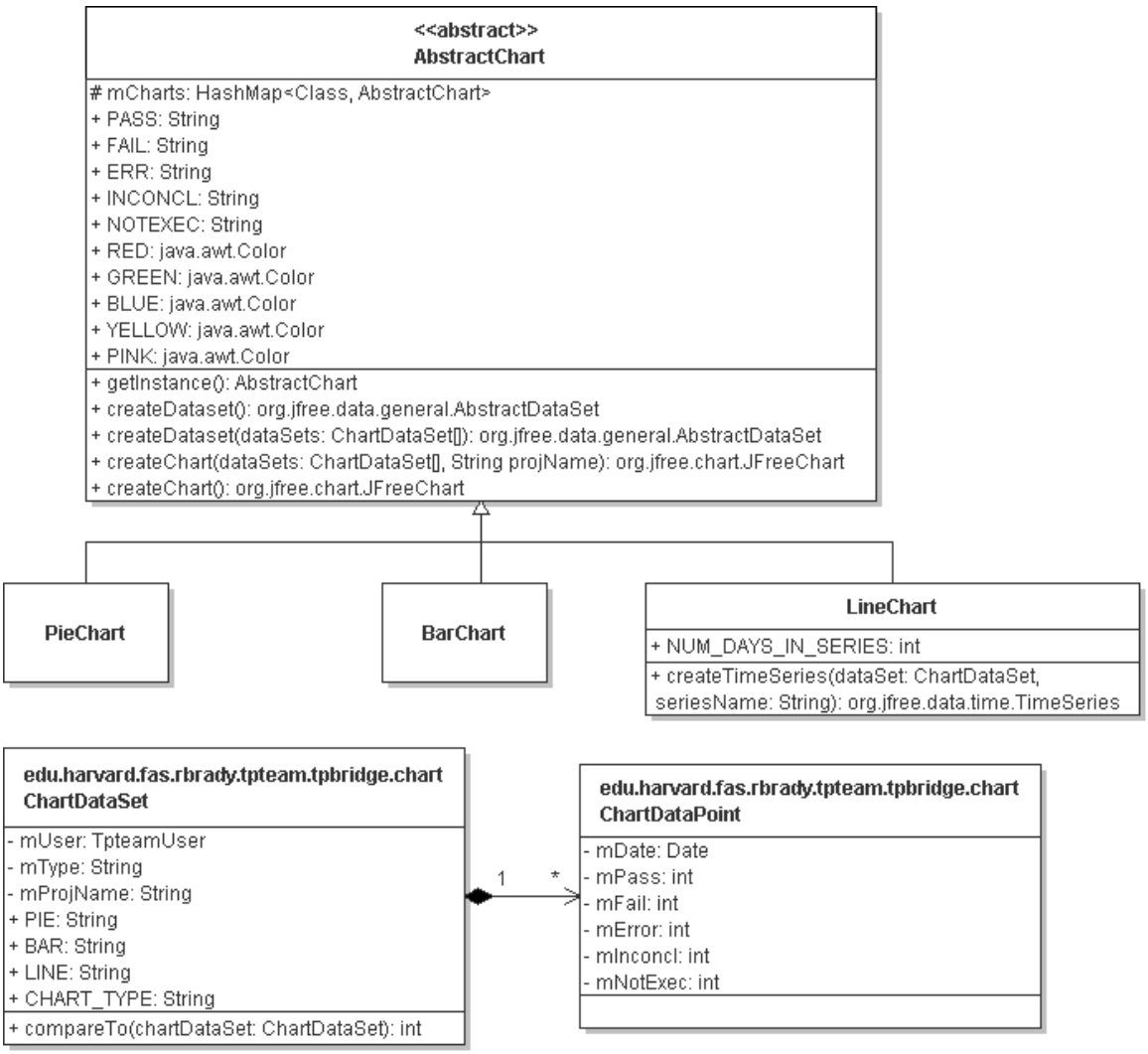

Figure 18 Package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts Class Diagram

at runtime through the inherited createChart methods.

Custom ChartDataSet and ChartDataPoint objects were created for TPTeam. A single data point represents a test execution verdict or set of verdicts run on a particular date. A data set is a composition of data points. However, a data set also contains properties describing the user who ran the test executions, the affiliated project, associated data points, and the chart type (Bar, Pie, or Line). A compareTo method was implemented for ChartDataSet so that data sets could be sorted by TpteamUser names.

Eventadmin package class diagram. Figure 19 below shows the class diagram for the eventadmin package. While the class names are identical to those found in the TPBridge eventadmin package, the components implement their interfaces differently and have a dissimilar hierarchy. The function of the two packages is clearly different.

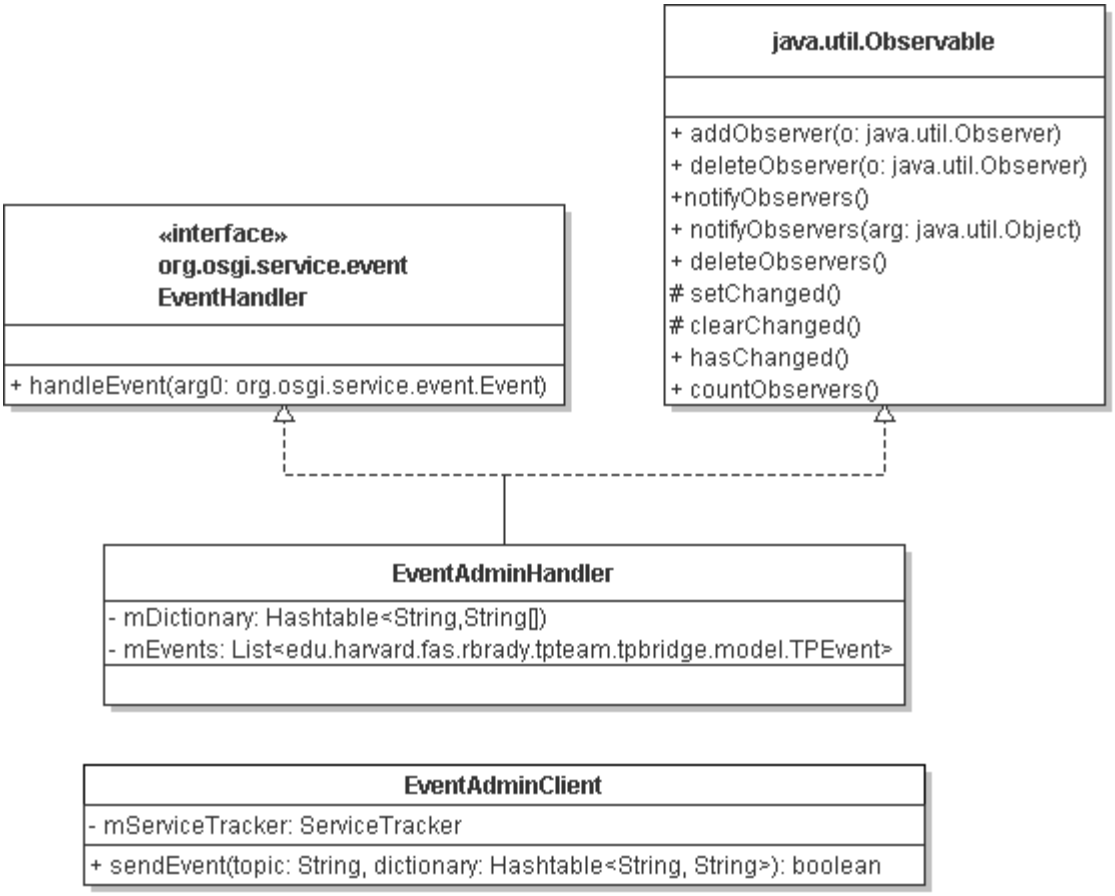

Figure 19 Package edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin Class Diagram

For example, while the TPBridge eventadmin package concerns itself only with inter-JVM messages, the TPBuddy eventadmin package only handles intra-JVM TPTeam messages. The TPBridge EventAdminHandler implements and Observer interface so that it can be notified when out-of-JVM messages arrive. The TPBuddy EventAdminHandler extends the Observable class so that it can notify the TPBuddy views when a TPEvent

has been issued within its own runtime. The TPBuddy EventAdminHandler does not have a private isTopicForwardable method because unlike TPBridge, it is not concerned with forwarding messages out-of-JVM.

Views package class diagram. The views package contains the classes that enable TPBuddy to display project details with graphic summaries, the test tree, test details, and the TPEvent history. All of the views are structured similarly:

- Inheritance: java.util.Observer is extended so view scan be notified when the EventAdminHandler receives a TPEvent of interest.
- Composition: custom content and label providers help to implement a Model-View-Controller pattern and keep views synchronized.
- Dependencies: implementations of org.eclispe.jface.action.Action carry out the Controller task for performing TPTeam Test CRUD and executions.

Since all of views are structured and behave similarly, only the TestView and directly related class will be shown here. They can be seen in Figure 20 below.

The diagram shows the TestView and its two custom helper classes: TestContentProvider and TestLabelProvider.

When the TestView receives a TPEvent update from the EventAdminHandler, it updates its member TreeNodeModel variable accordingly. This in turn causes the TestContentProvider to synchronize the view. The TestLabelProvider provides the means to read the view's data model and assign the appropriate image (folder, test, execution verdict) and text to nodes within the test tree.

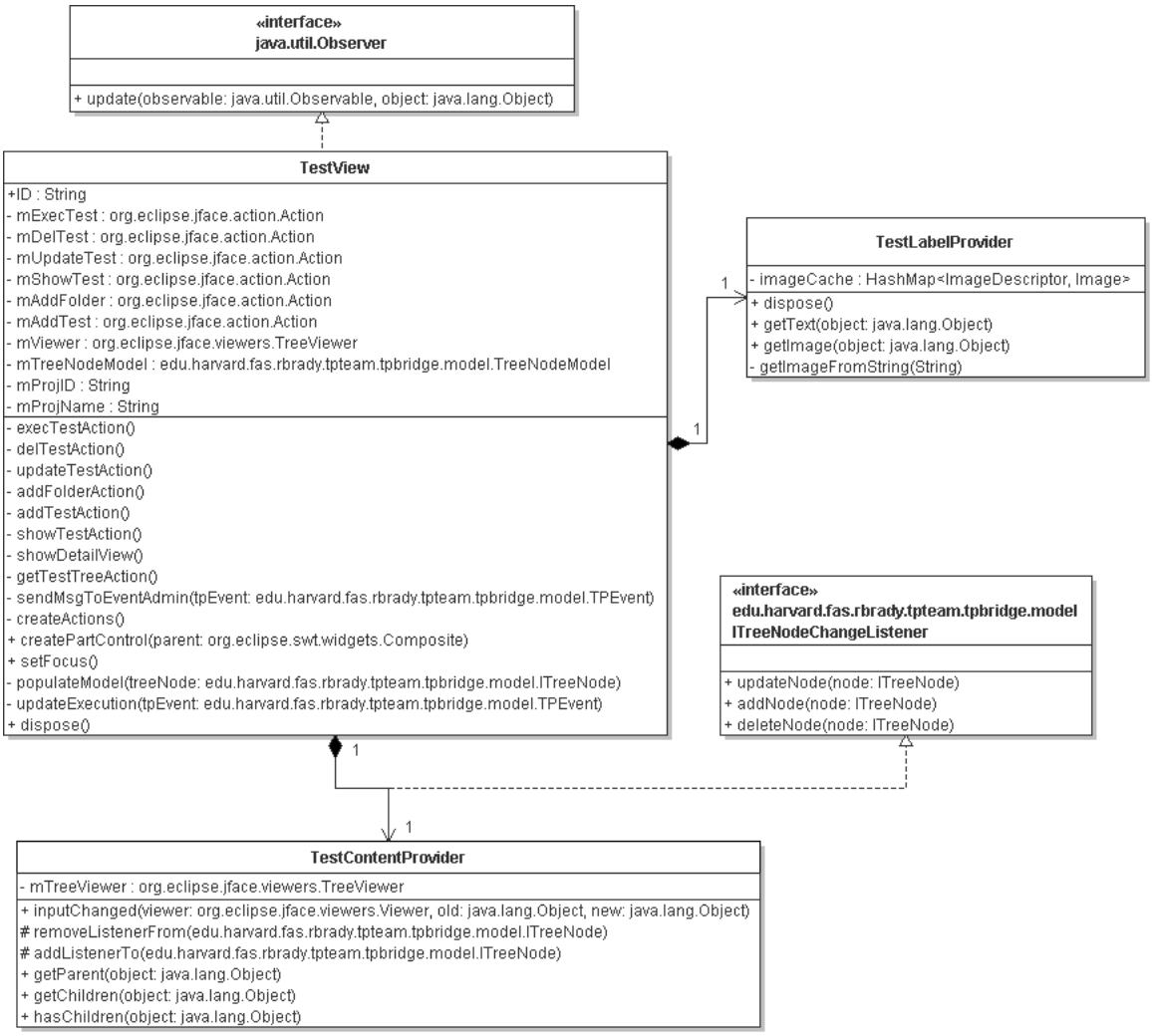

Figure 20 Package edu.harvard.fas.rbrady.tpteam.tpbuddy.views Class Diagram, TestView

TPManager

The TPManager plug-in consists of sixteen packages. They are listed in Table 12 below. The significant package in terms of object model design is tptp. This package provides multi-threaded response to incoming requests from both the TPBridge and Web clients.

TPManager Package	Functionality
NA (base)	Holds OSGi Activator class that controls the starting and stopping of the plug-in as well as initialization of TPBridge client and OSGi event handler/client.
eventadmin	OSGi event messaging classes.
hibernate	Utility classes for performing CRUD through Hibernate framework.
http.admin.*	Servlets to expose TPTeam CRUD to administrative Web clients.
http.user.*	Servlets to expose TPTeam CRUD to non-administrative Web clients.
tpbridge	Provides the client to the TPBridge exposed OSGi service.
tptp	Implements business logic upon incoming TPEvents received from ECF or HTTP requests.

Table 12 TPManager Packages (edu.harvard.fas.rbrady.tpteam .tpmanager.* Prefix)

The class diagram of TPManager's tptp package is given in Figure 21 below. Here it can be seen that the TPManager class extends java.util.Observer. This is done so that it can be notified in real time when an out-of-JVM TPEvent is received by its TPBridge. Upon receiving a CRUD or execution request, a TPManager object instantiates a TPManagerThread to handle the request. The thread determines which type of CRUD or execution request was issued (via the event topic) and calls the appropriate synchronized method of TPTestCRUD or TPTestExec.

It was a design decision to allow both CRUD and execution threads to run in parallel. The reason was to provide better response to the TPBuddy GUIs. Test executions can take tens of seconds to complete. Test CRUD takes at most on the order

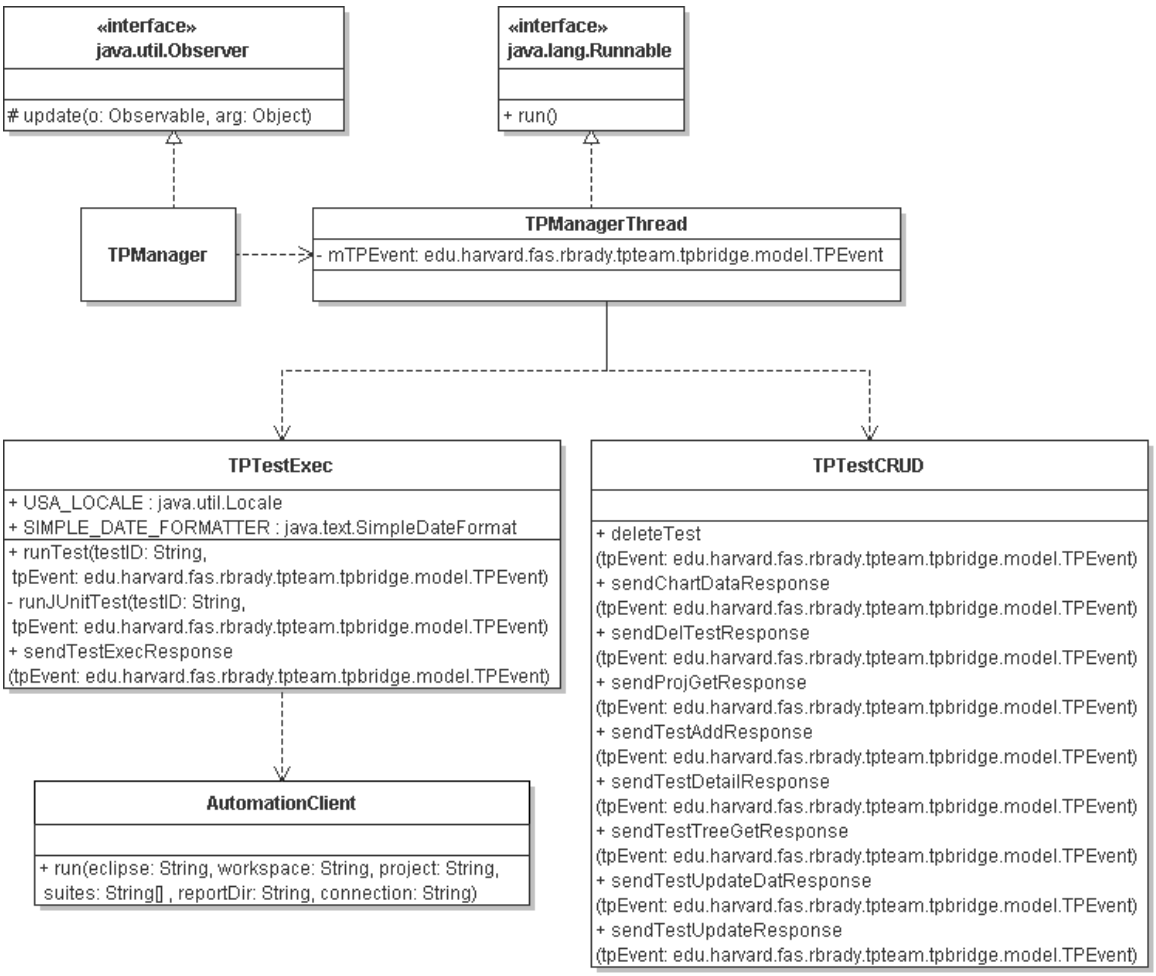

Figure 21 Package edu.harvard.fas.rbrady.tpteam.tpmanaager.tptp Class Diagram

of seconds to complete. It would be unacceptable for TPTeam users to have to potentially wait tens of seconds in order to complete simple CRUD operations. It is the author's experience that users expect and are willing to tolerate remote test executions to take much longer than simple test project management metadata operations. Hence, both CRUD and executions are allowed to take place in parallel on the TPTeam system.

Relational Model

The relational model was derived from the Hibernate object model given in Figure 16. The Hibernate tools package, discussed in Chapter 2, was used to generate the database schema shown in Figure 22 below. Hibernate correctly generated the linking table PROJ_USER. This was done in order to split the many-to-many relation between the PROJECT and TPTEAM_USER tables in to two one-to-many relations.

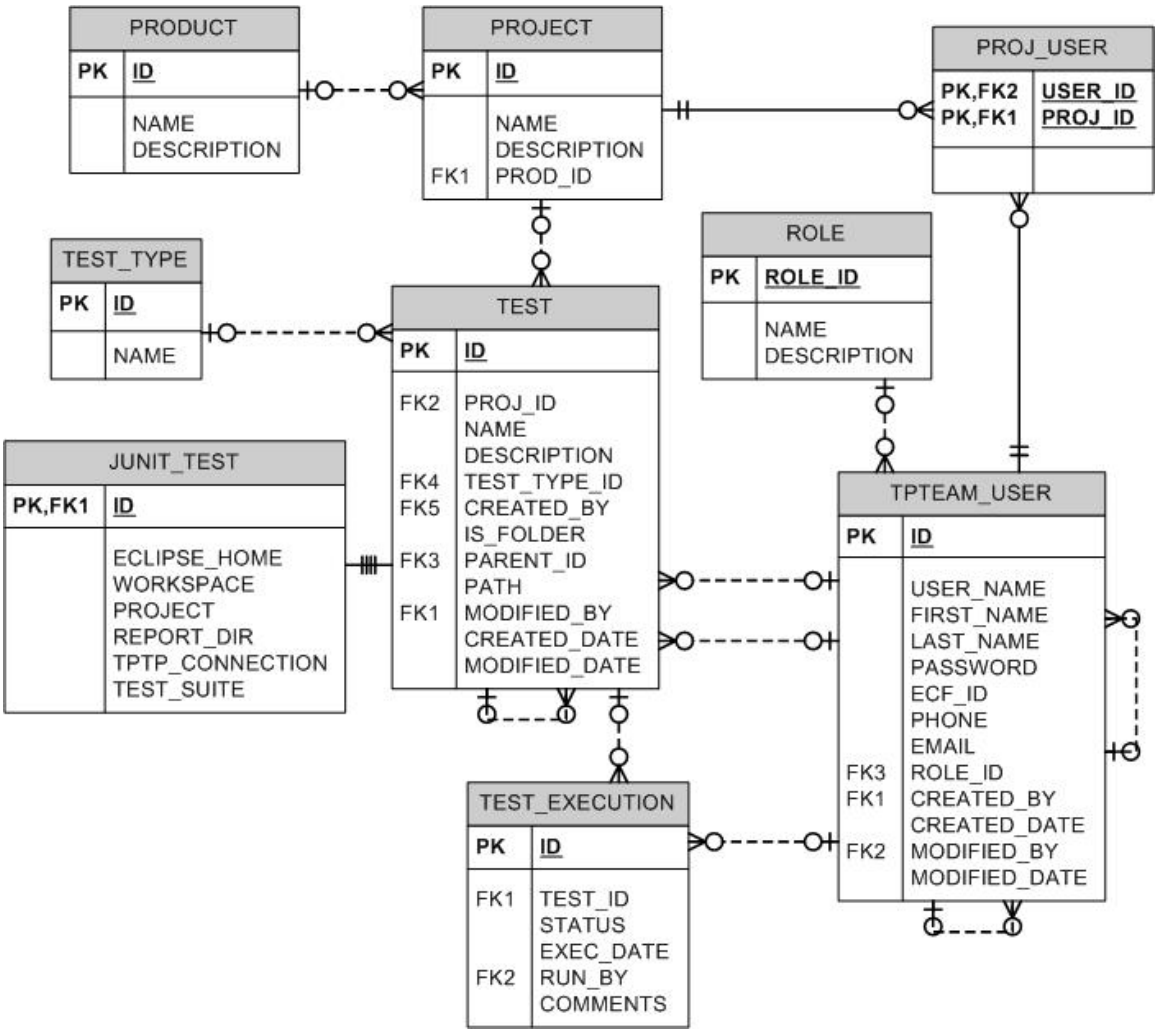

Figure 22 TPTeam Database Relational Model

The author generated a normalized database schema (Powell, 2005) by hand to see if the results would be different from those produced by Hibernate. The results were identical.

This chapter provided an overview of the system architecture, object model, and relational design of TPTeam. The following chapter will show the reader how to access the source code and binaries, install, configure, and finally use TPTeam.

Chapter 4 User Guide

This chapter provides the reader with the information necessary to download, install, configure, and run TPTeam. The project is supported by two code hosting sites, Google Code (Google, 2007) and Sourceforge.net (Sourceforge, 2007). Each site has links to project background information, source and example code, bug submission forms, community forums, etc. Table 13 below summarizes the features of the two sites.

TPTeam Website	Features
http://code.google.com/p/tpteam	Current code repository. Provides release downloads ready to be unpacked and deployed. Hosts a project wiki and issues reporting/tracking tool open to the public. Google tpteam-dev group available for discussion.
http://sourceforge.net/projects/tpteam	Legacy code repository to support EclipseWorld 2006 presentation. Provides tutorial presentation slides aimed to incrementally introduce reader to TPTeam technologies. Hosts self-contained example plug-ins that act as “hands-on” development companions to tutorial presentation.

Table 13 TPTeam Code Hosting Websites Summary

The above websites can be used a companion reference to this report.

Access to Source Code and Binaries

The source code and binaries suitable for deployment are hosted on the Google website given above in Table 13. Detailed instructions on how to use the website as a repository and how to download the binaries are given by Google and won't be covered here.

The Google hosted source code is stored in a Subversion (CollabNet, 2007) repository. Any standard subversion client may be used to anonymously download the code. Binary releases of the TPTeam components are also available at the Google website. They can be downloaded via an ordinary Web browser. Table 14 below summarizes the current binary downloads available.

Binary File	Summary
tpteam_plugin_deps_0.1.zip	Contains all third-party plug-ins required by all TPTeam components. README instruction file is included.
tpbuddy_0.1.zip	The complete TPBuddy RCP application, including TPBuddy and TPBridge plug-ins, but not the required third-party ones. README instruction file is included.
tpmanager_webapp_0.1.zip	The complete TPManager Equinox bridge Web application. Includes the TPManager and TPBridge plug-ins, but not the third-part ones. Packaged in a Tomcat Web application directory structure. README instruction file is included.

Table 14 TPTeam Binary Release Summary

Installation

The installation of the TPTeam components is a straightforward process. Once the binary distribution is downloaded and unpacked, only a few configuration files need be changed and the database seeded with data. These operations are covered in the next sections.

System Requirements

The system requirements for the installation and running of TPTeam are given in Table 15 below.

Requirement Type	Acceptable Values
Operating System	Windows XP, Service Pack 2
Sun Java Runtime	Version 1.5.11 or later
Random Access Memory	512 MB minimum, >= 1 GB desirable
CPU Speed	1 GHz minimum
Relational Database	MySQL 5.0, Oracle XE 10.2
Servlet Container	Any Servlet Specification 2.4 compliant container. Apache Tomcat version 5.5 used by author.
XMPP Accounts	Any combination of accounts from XMPP providers is acceptable. Gmail preferred by the author for best uptime.

Table 15 TPTeam System Requirements

TPTeam requires basic system administrator skills in order to install the prerequisite Web application and database servers. It is assumed the person doing the TPTeam installation already possesses these skills.

TPBuddy

Download and unzip the tpteam_plugin_deps and tpbuddy zip file releases. Copy all of the plug-ins from the tpteam_plugin_deps plugins directory to the unzipped tpbuddy plugins directory.

TPManager

Download and unzip the tpmanager_webapp zip file. Copy the contents of the unzipped webapps > bridge directory into the Web application directory of the servlet container to be used as a host for TPManager. Additionally, if needed, copy the appropriate Java database driver from the unzipped common > lib directory into the shared library directory of the Web application server to be used. Drivers for the MySQL and Oracle XE databases are provided.

Create the TPTeam database schema by running the tpteam_ddl.sql and tpteam_dml.sql scripts located in the unpacked tpmanager_webapp sql directory. The tpteam_ddl script contains the SQL data definition language commands necessary to create the TPTeam database tables, primary key, and foreign key constraints. The tpteam_dml script inserts seed data so that initial security roles, administrative user, and other tpteam metadata are created. The above SQL scripts can be run from the command line using the appropriate MySQL or Oracle XE shell utility.

Configuration

The configuration of the TPTeam components is a relatively simple task. It mostly involves only changing a few plain text configuration files for logging, database

connections, and instant messaging. The following sections describe how to perform these operations.

TPBuddy

TPBuddy needs only one text property changed for its configuration, `tpmanager.ecfID`. This property is located inside file tpteam.properties located under the TPBuddy unpack directory at plugins > edu.harvard.fas.rbrady.tpteam.tpbridge_1.0.0 > data. Change the value to the XMPP address of the TPManager instance, e.g., "tpmanager@gmail.com".

TPManager

The TPManager Web application needs a few configuration file properties set so that it can connect to its database, log operations, connect to XMPP instant messaging, and enforce Web security. Table 15 below summarizes how to set the appropriate values.

An example server.xml configuration file for the Servlet Container to be used is provided in the conf directory of the distribution. It provides an example configuration of the JDBCRealm security tag that is used by the Tomcat Web server. This configuration is based upon the default TPTeam database table schema. The use of this configuration allows Tomcat to manage the login sessions of the TPTeam Web users.

Configuration Property	How to Set Appropriate Value
`tpmanager.ecfID, tpmanager.password`	Properties are located inside file tpteam.properties located under the TPManager Web application directory at plugins > edu.harvard.fas.rbrady.tpteam.tpbridge_1.0.0 > data. Change the values to the XMPP address of the TPManager plug-in, e.g., "tpmanager@gmail.com". Change the password to its correct value.
`hibernate.connection.*: driver_class, url, username, password, default_schema`	Properties are located inside file hibernate.cfg.xml located under the TPManager Web application directory at plugins > edu.harvard.fas.rbrady.tpteam.tpbridge_1.0.0. Change the values to those appropriate for the database driver, connection, and schema to be used.
`log4j.appender.file.File`	Property is located inside file log4j.properties located under the TPManager Web application directory at plugins > edu.harvard.fas.rbrady.tpteam.tpbridge_1.0.0. Change the value to the file location where Hibernate operation logging should be kept.
`<JDBCRealm> Security Realm`	Property located inside server.xml file of the Servlet Container used. Values based upon the default TPTeam schema are provided in the file conf > sample_server.xml of the TPManager distribution.

Table 15 TPManager Configuration Property Settings

Running the TPManager Server

This section describes how to run TPManager as an Eclipse runtime embedded within a Web server. It assumes that the installation and configuration steps for TPManager have already been performed.

Once TPManager has been installed and configured, running it is a straightforward process. First, be sure the TPTeam database has been created and seeded with data from the SQL scripts mentioned in the TPManager installation section. Next,

start the database. Finally, start the Web server that where TPManager was installed as a Web application. TPManager will print the following items to standard output during startup: the topics of all TPTeam events its TPBridge will send, an indicator that it has registered with the OSGi Event Admin Service, and an activation status of its TPSharedObject.

During the installation of TPManager, an administrative user with username "tpteam1" and password "tpteam" was created. This initial administrative user can be used to access TPManager from a Web browser and add more users, products, projects, etc. A discussion of how to perform these operations via a Web browser is given in a following section.

Running TPTeam: TPBuddy RCP Application

This section describes how to carry out TPTeam operations through the TPBuddy RCP application. A following section will describe how to conduct the same operations through the Web interface. It is assumed that a TPManager instance is already running and available for exchanging TPTeam messages.

Logging In/Out

Click on the tpbuddy.exe file in the directory where the tpbuddy zip file was unpacked. The RCP application GUI should appear on your screen. Click on the person icon in the upper left corner of the GUI. The XMPPS connection wizard dialog as shown in Figure X below will pop up on your screen. Enter your XMPPS user ID and password, then click the finish button on the wizard.

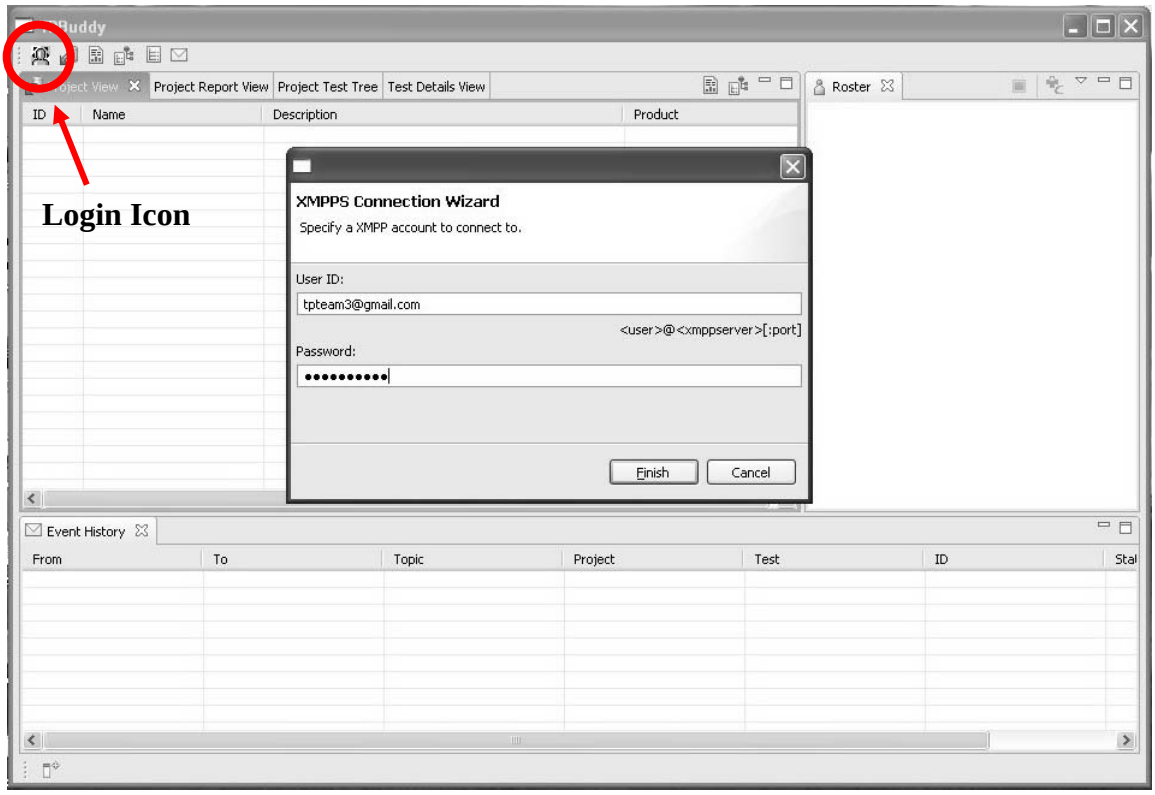

Figure 23 TPTeam RCP Login Dialog

Once logged-in, the roster view in upper right hand side of the TPBuddy GUI should display a tree of other TPBuddy peers online. In particular, the TPManager should be visible as a connected peer. Figure 23 below shows an example of the TPBuddy GUI right after a successful login. Additionally, the project view will be populated with any project which the user has membership. This can also be seen in Figure 24. Finally, since the population of the project view consisted of the issuing of a "project get request" (projgetreq) by TPBuddy and a "project get response" (projgetresp) by the TPManager, these events are logged in the event history view.

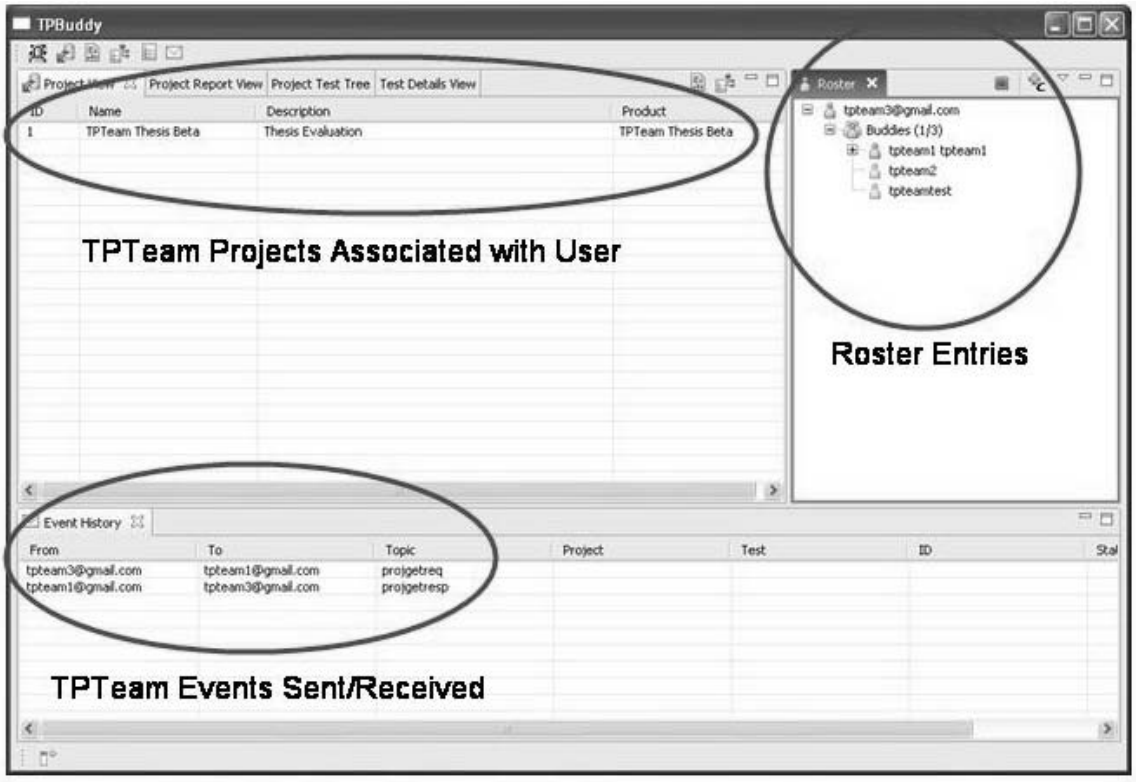

Figure 24 TPBuddy GUI After Successful Login

A user can log out from an existing XMPPS connection by simply clicking on the red square located in the upper right hand part of the roster view. This red logout square is also shown in Figure 24.

TPTeam Database Get Test Plan Tree

After logging into the system, the TPBuddy user can view the test plan tree for a specific project. This is done by highlighting a project's row by clicking on it in the project view and then clicking on the test tree icon at the top of the view. This is shown in Figure 25 below.

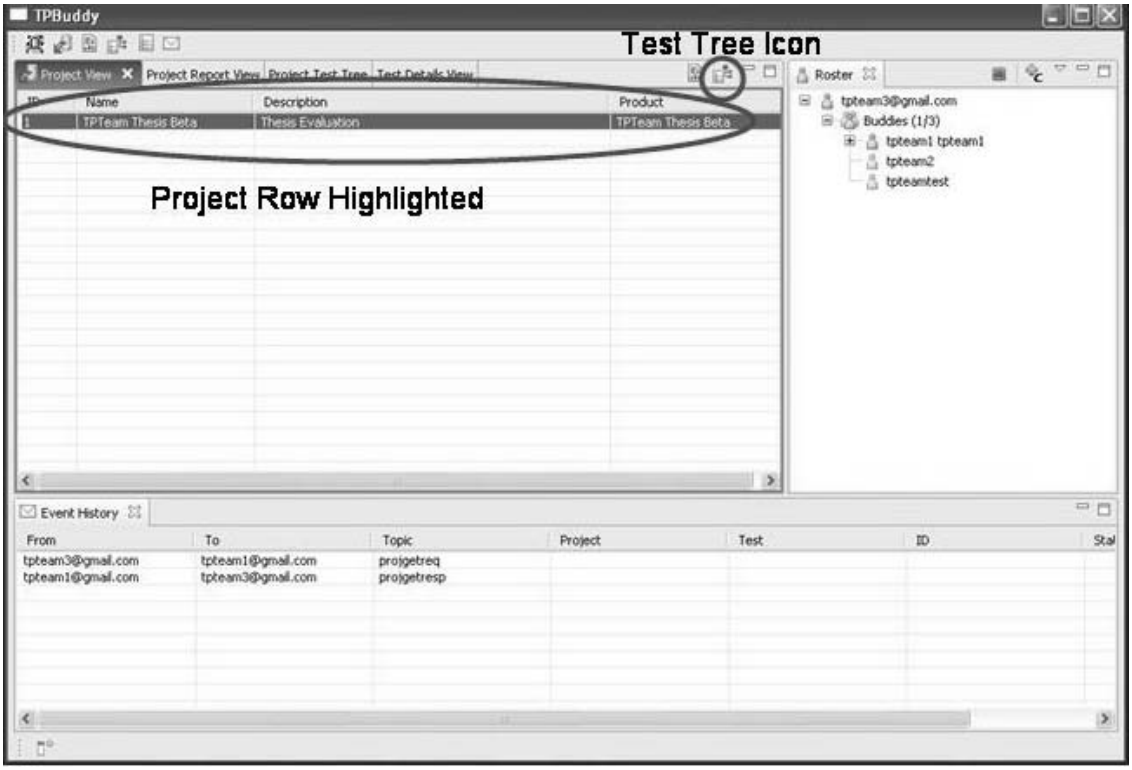

Figure 25 Project Row Highlighted for Test Tree Selection

After the test tree icon is clicked, the test tree view will automatically get focus and be populated with nodes. The user can then browse the test tree in a fashion similar to directory explorer browsing in Windows XP. Figure 26 shows a tree view partially expanded for browsing. Also shown in Figure 26 is the test tree view toolbar. The toolbar contains action icons for executing a test, adding a test folder or definition, updating a tests, viewing test details, and finally, deleting a test. These features will be discussed next.

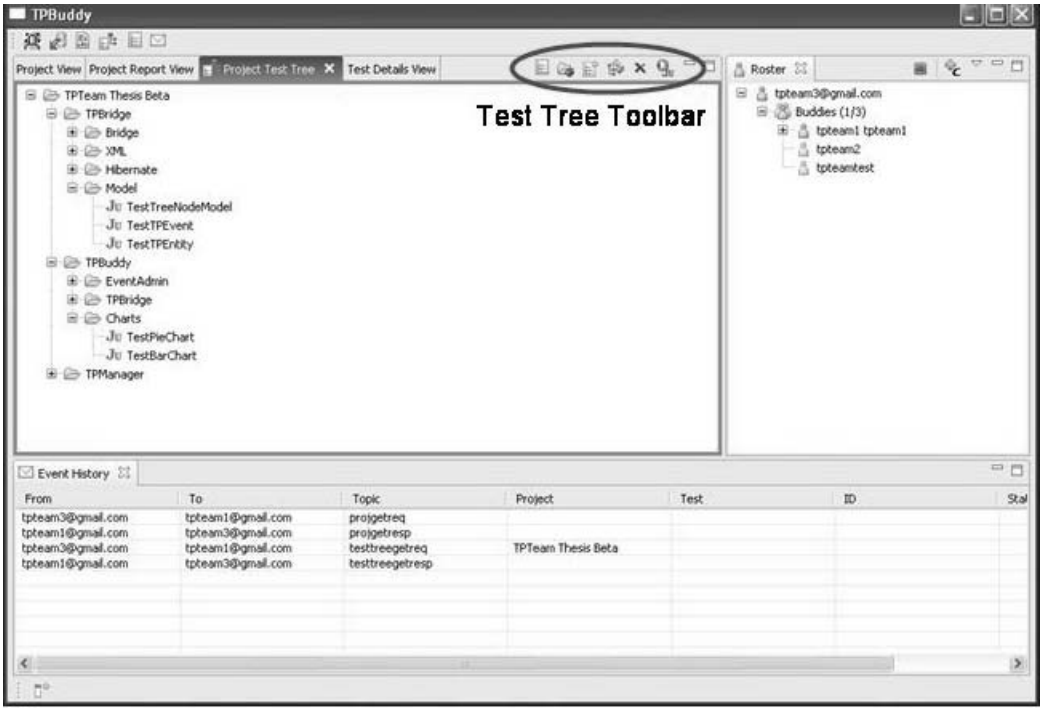

Figure 26 Test Tree View

Execute Test Definition

Test executions are carried out in a similar way as how a project was selected before the test tree could be viewed. In both cases, first an item is highlighted by clicking on it and then an action icon in the view is selected. A test execution is performed by first highlighted the desired JUnit test in the tree and then clicking on the green triangular test execution icon in the view.

When the TPManager has finished collecting the test verdict and the results TPEvent has been received by the bridge of TPBuddy, the execution will be displayed in the tree in real time. Figure 27 below shows this scenario.

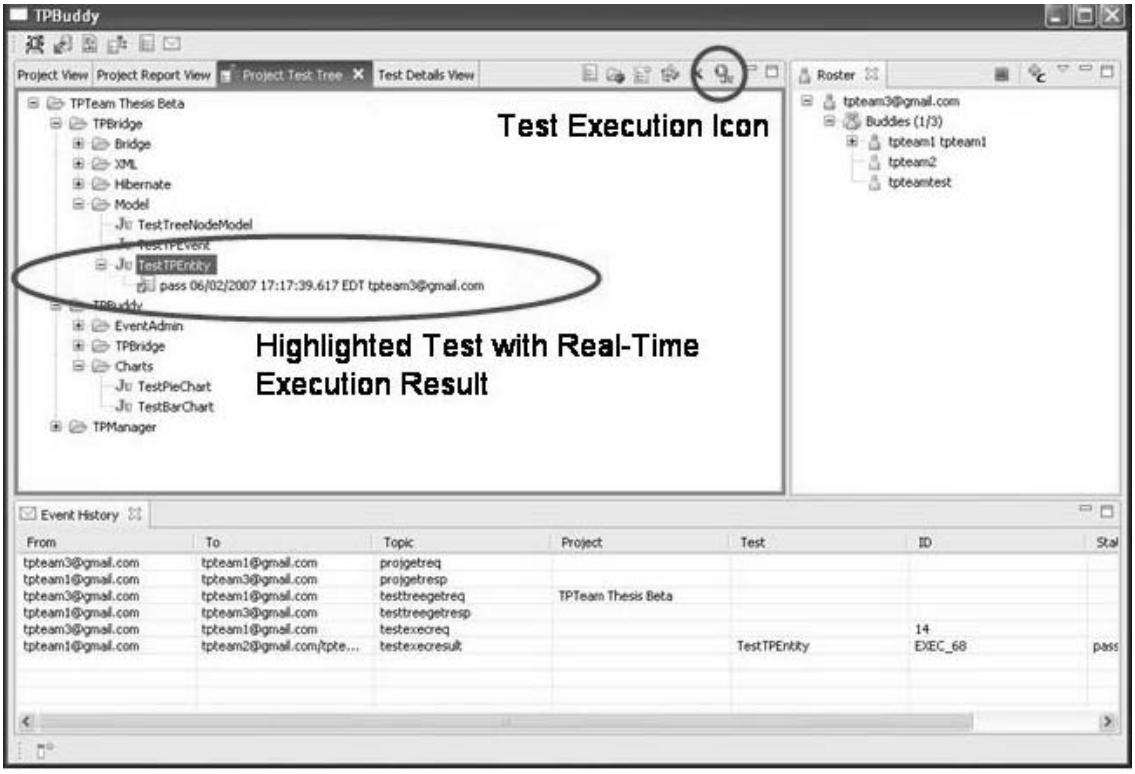

Figure 27 Test Execution in TPBuddy

Add and Update Test Folder/Definition

Test folders and definitions are added in a similar way. In each case, the parent folder is first highlighted and then the "add folder" or "add test definition" icon is selected. Once an add action icon is selected, an input dialog will appear, prompting the user to add the test entity information. Once all the required information is entered and the ok button pressed, TPManager will store the information in the database. The new test tree node will appear in real time. Figure 28 below shows the add test definition dialog.

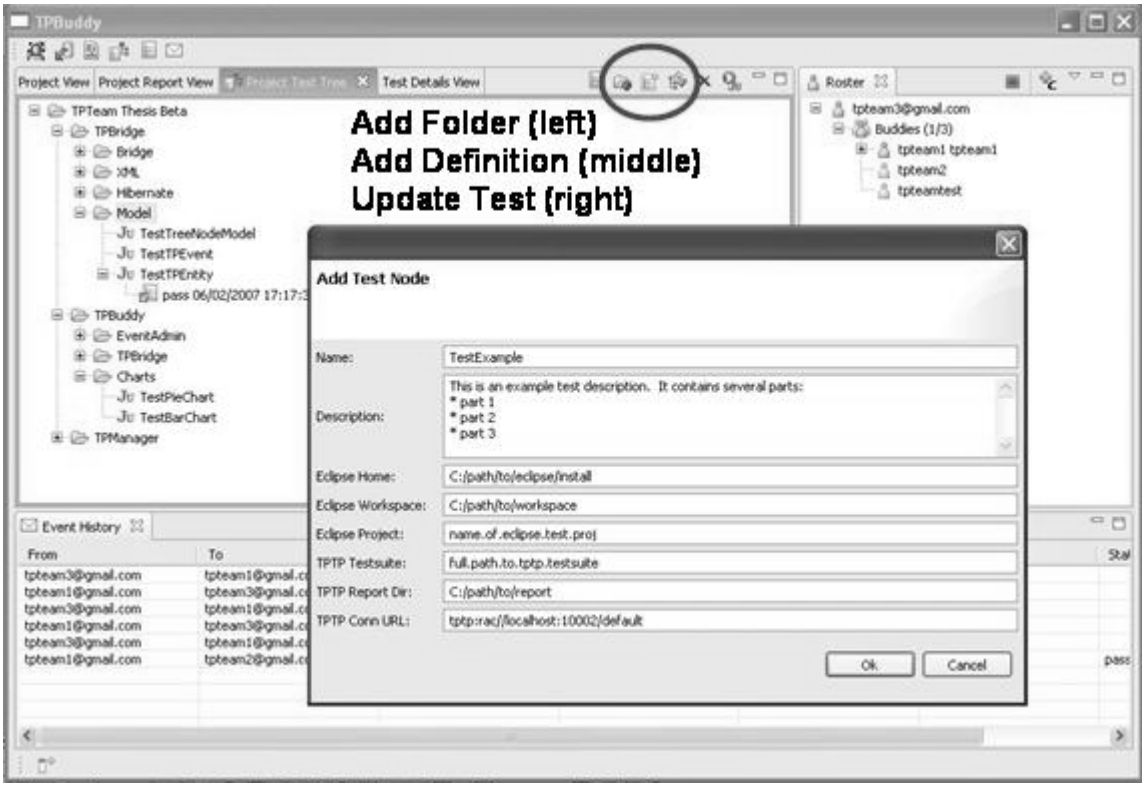

Figure 28 Add Test Dialog

The updating of folders and definitions works in the same way. The difference is that a single update icon is used for either a folder or definition. The update icon is shown at the far right in Figure 28. First the folder or definition to be updated is highlighted and then the update icon selected. The appropriate dialog will then appear, fully populated with the current information stored in the database. The user need only edit the formation and then on the dialog's ok button to persist the changes.

Delete Test Folder/Definition

The deletion of folders and definitions occurs similarly to the other operations described above. First, the test folder or definition to be deleted is selected in the tree view. Secondly, the red "x" action icon shown in Figures 26-28 is selected. Next, a

confirmation dialog will appear, asking the user is he is sure to proceed with deletion. Finally, once the dialog's ok button is pushed, the TPManager will delete the definition and all its execution information from the database. In the case of a folder deletion, all child folders and test definitions with their executions will be deleted. The appropriate TPEvents will be exchanged and the test tree view updated in real time.

View Test Folder/Definition Details

Detailed information about each test folder or definition can be obtained from the TPBuddy test detail view. Not only does it list metadata about tests, but it also displays the execution history of definitions.

First, a test folder or definition is selected in the test tree view. Next, the test details action icon is selected in the test tree view. The details action icon is the leftmost one in the view's toolbar. Finally, the test details view should automatically be given focus and be populated with all pertinent information from the database. Figure 29 below shows an example of the test details view in action.

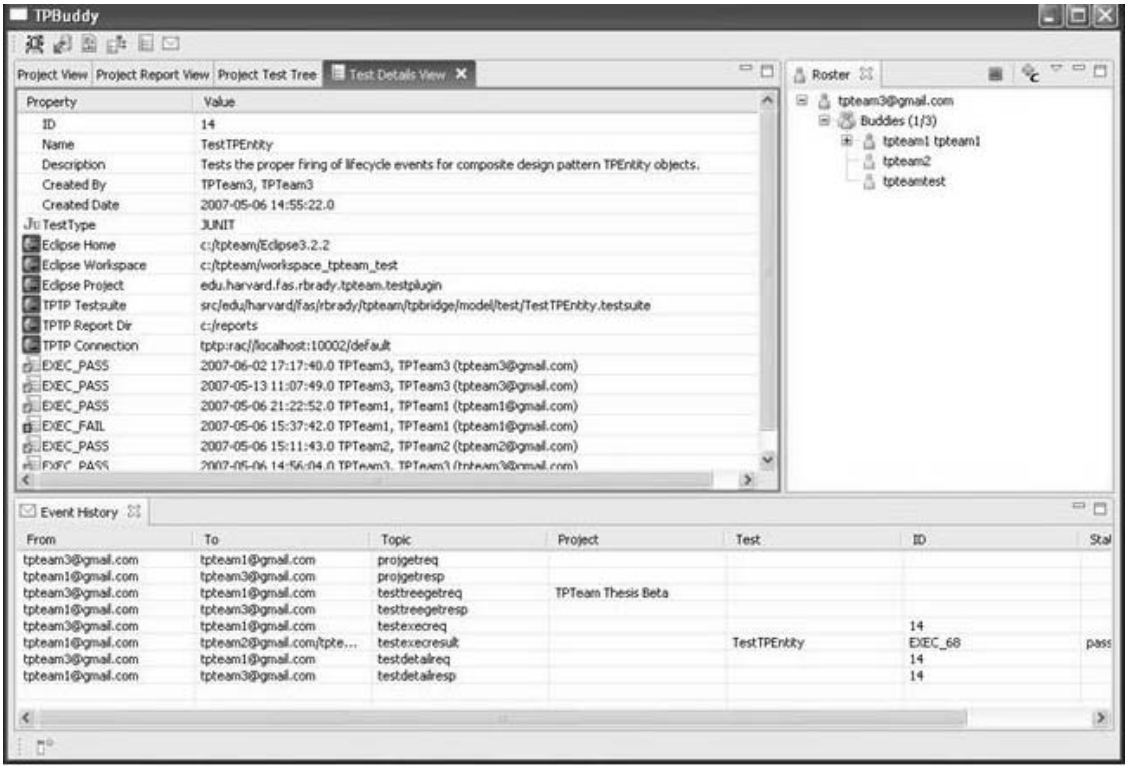

Figure 29 Example Test Detail View

View Project Graphs

TPTeam also provides the means to graphically view test project progress. The project report view allows a user to view a pie chart of overall test plan results, a bar chart of the test status of individual developer contributions, and finally, a line graph of the time progression of the project. The following Figures 30, 31, and 32, show each of these graphs, respectively.

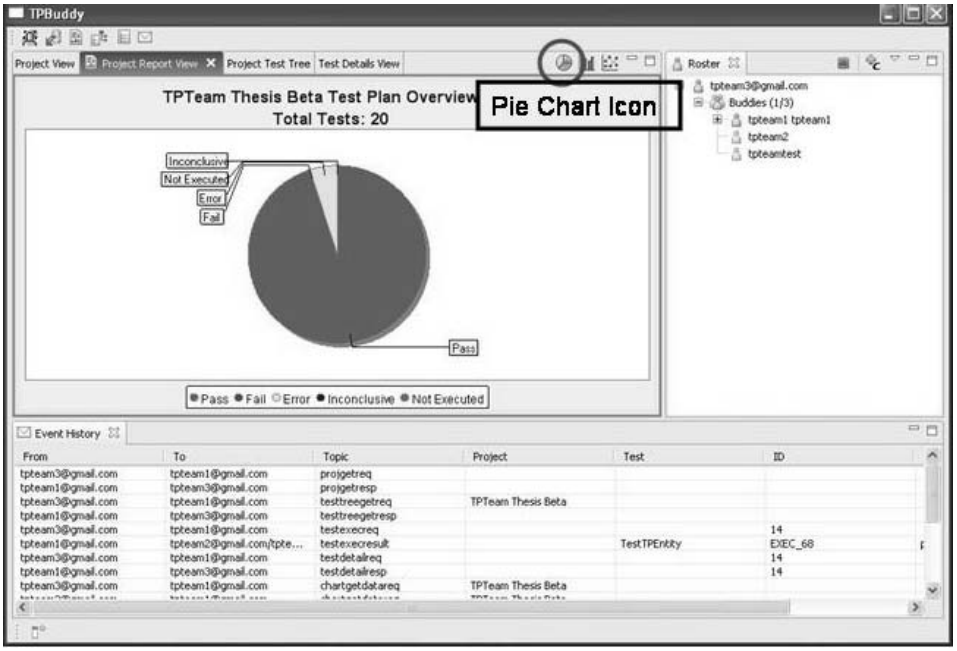

Figure 30 TPBuddy Pie Chart View

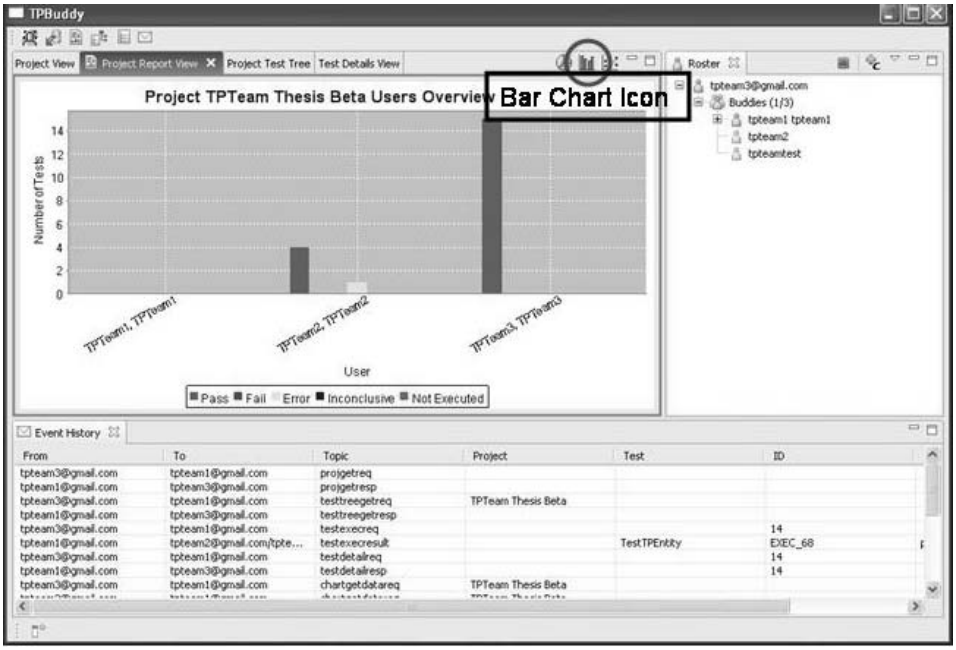

Figure 31 TPBuddy Bar Chart View

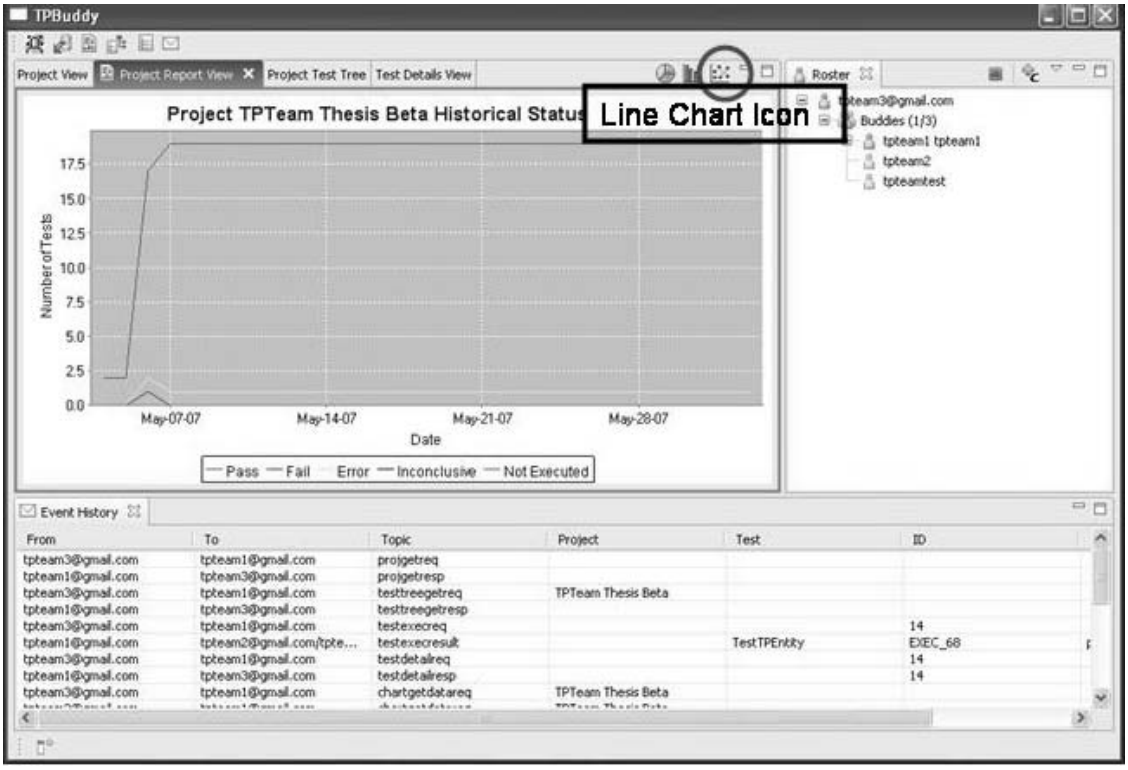

Figure 32 TPBuddy Line Chart ViewTPBuddy Peer Instant Messaging

TPBuddy Peer-to-Peer Instant Messaging

The TPBuddy RCP GUI can also be used to communicate via XMPP instant messaging with its peer TPBuddies. This is done by first right-clicking on the desired peer in the Roster View tree. Next, select the "Send IM to Peer" item from the context menu. An instant messaging chat dialog should then appear. Figure 33 below shows an example of a TPBuddy peer highlighted and the resulting instant messaging dialog in action. The top portion of the dialog maintains a scrolling transcript of the chat session. The bottom portion is where outbound messages are entered. The scrolling transcript is updated in real time as messages are sent and received.

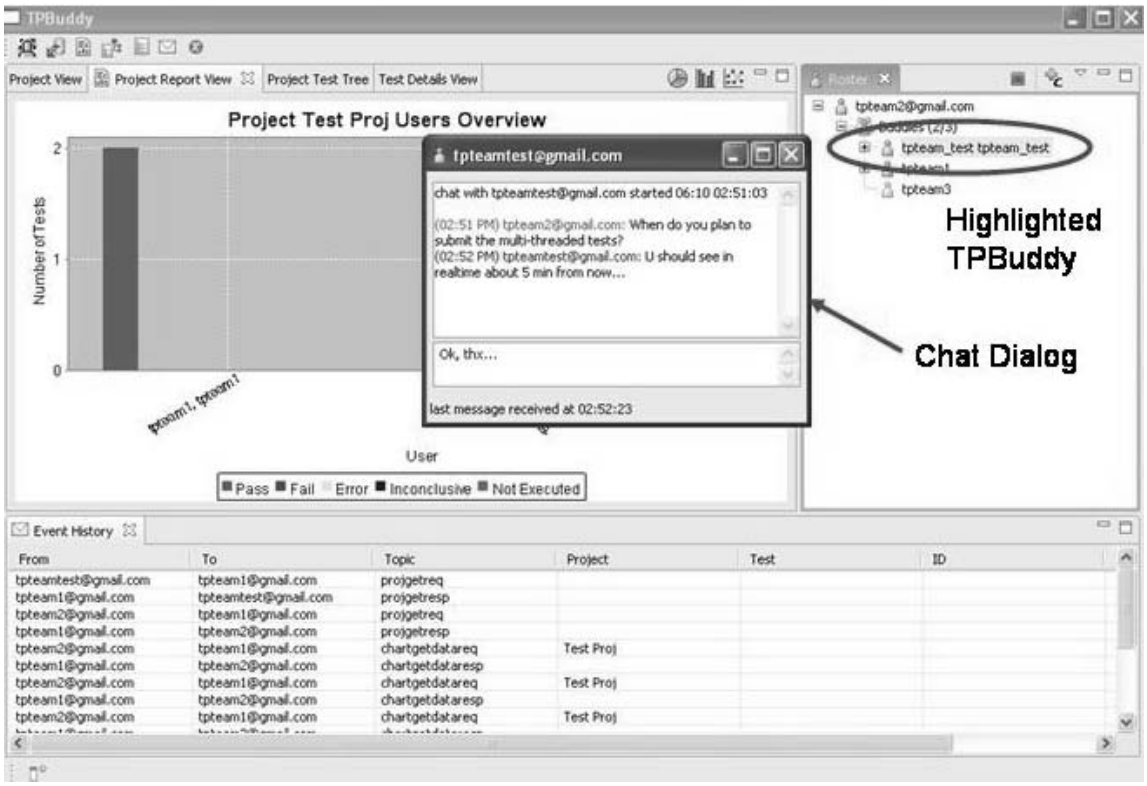

Figure 33 TPBuddy Instant Messaging Chat Dialog

Running TPTeam: Web Application

TPTeam also exposes its functionality through a Web interface. It has an administrative set of webpages for performing CRUD operations on products, projects, users, and tests. There is also a convenience set of pages for users to perform CRUD and execution operations solely on the tests or their own user account. Table 16 below summarizes the default install location of the TPManager Web interface.

The following discussion summarizes how to use the Web interface. Since the user set of pages is effectively a subset of the administrative ones, this discussion will center around the administrative pages.

TPManager Website URL	Features
http://localhost:8080/bridge/tpteam/admin/index.html	Administrative CRUD operations on Products, Projects, Users, and Test Folders/Definitions.
http://localhost:8080/bridge/tpteam/user/index.html	User CRUD and execution operations on Test Folders/Definitions they created, plus their own user accounts.

Table 16 TPManager Web Interface Summary

Choosing an Operation on a Type

After logging into the system by entering their username and password at the index.html pages in Table 16 above, the interface displays the home page. The home page presents two lists: one for selecting a CRUD or execute operation, and one for selecting the type of object to be the operation target. Figure 34 below shows the lists expanded.

Once an operation and target type are selected, the user is directed to either a form page for data input or a test tree widget view for test operations. These operations will be discussed in the following sections.

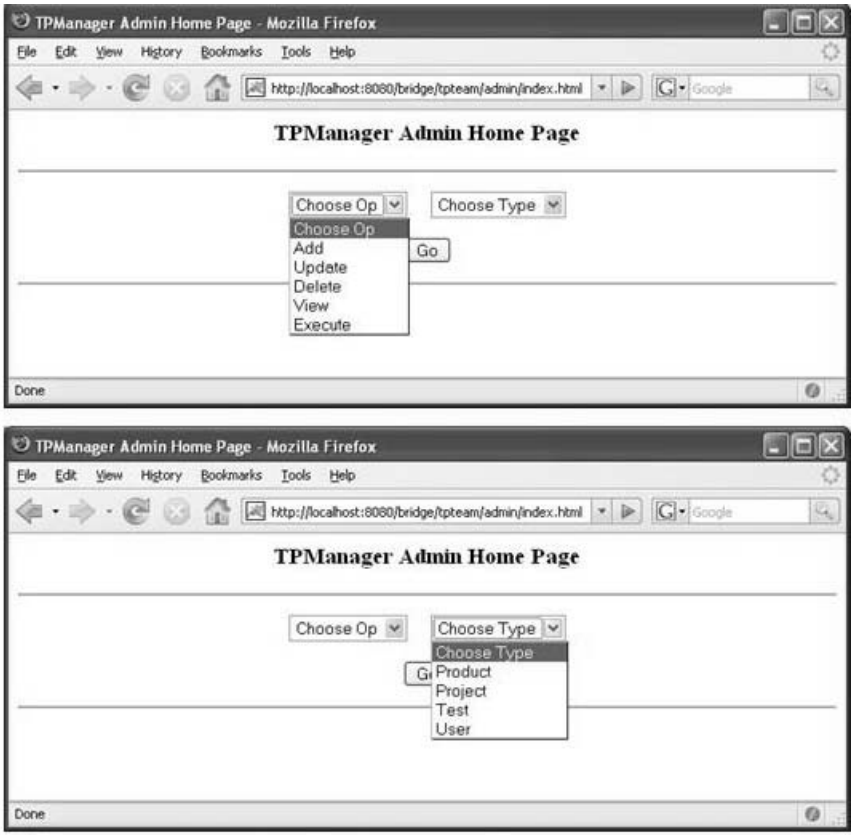

Figure 34 TPManager Home Page

Adding/Updating/Viewing a Product, Project, or User

The addition, update, and viewing of a product, project, or user are done similarly. In the addition case, the user is presented with a form to fill out. When applicable, the form contains one or more lists in order to make the desired associations with other objects already persisted in the database. For example, the add user form contains text boxes for the user name and other independent information. However, the form displays a list box for TPTeam role (e.g., “admin” or “user”) and also for associated projects. The form is given below in Figure 35.

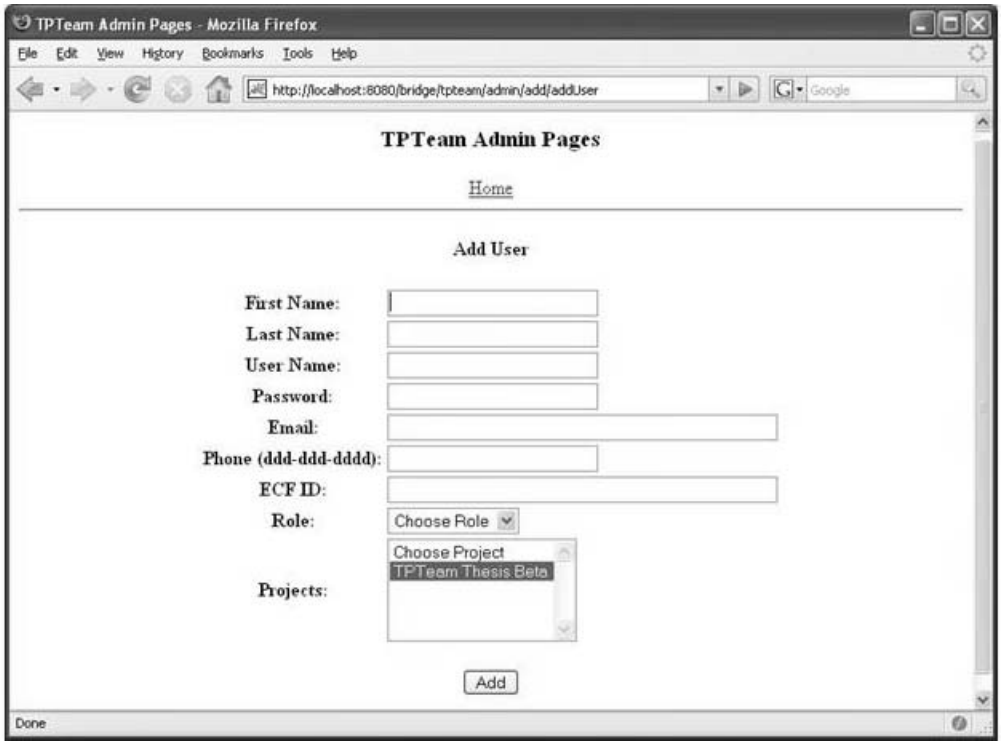

Figure 35 TPManager Add User Webpage

The update operations forms are identical to the addition ones. The only difference in the procedure is that initially the interface presents a tabulated list of the objects available for update. Once an object from the list is selected, a form populated with the object's current data is shown. The data in the form can then be edited and saved to the database. Viewing object details is done with a procedure nearly identical to updates, the difference being that the data populated forms are non-editable.

Adding/Updating a Test Folder or Definition

Test folders and definitions are added and updated with the help of a JavaScript tree widget developed by the author. Webpages for these operations contain textual form elements plus a test tree widget for selecting the parent node. An example of adding a JUnit test to the project is given in Figure 36 below.

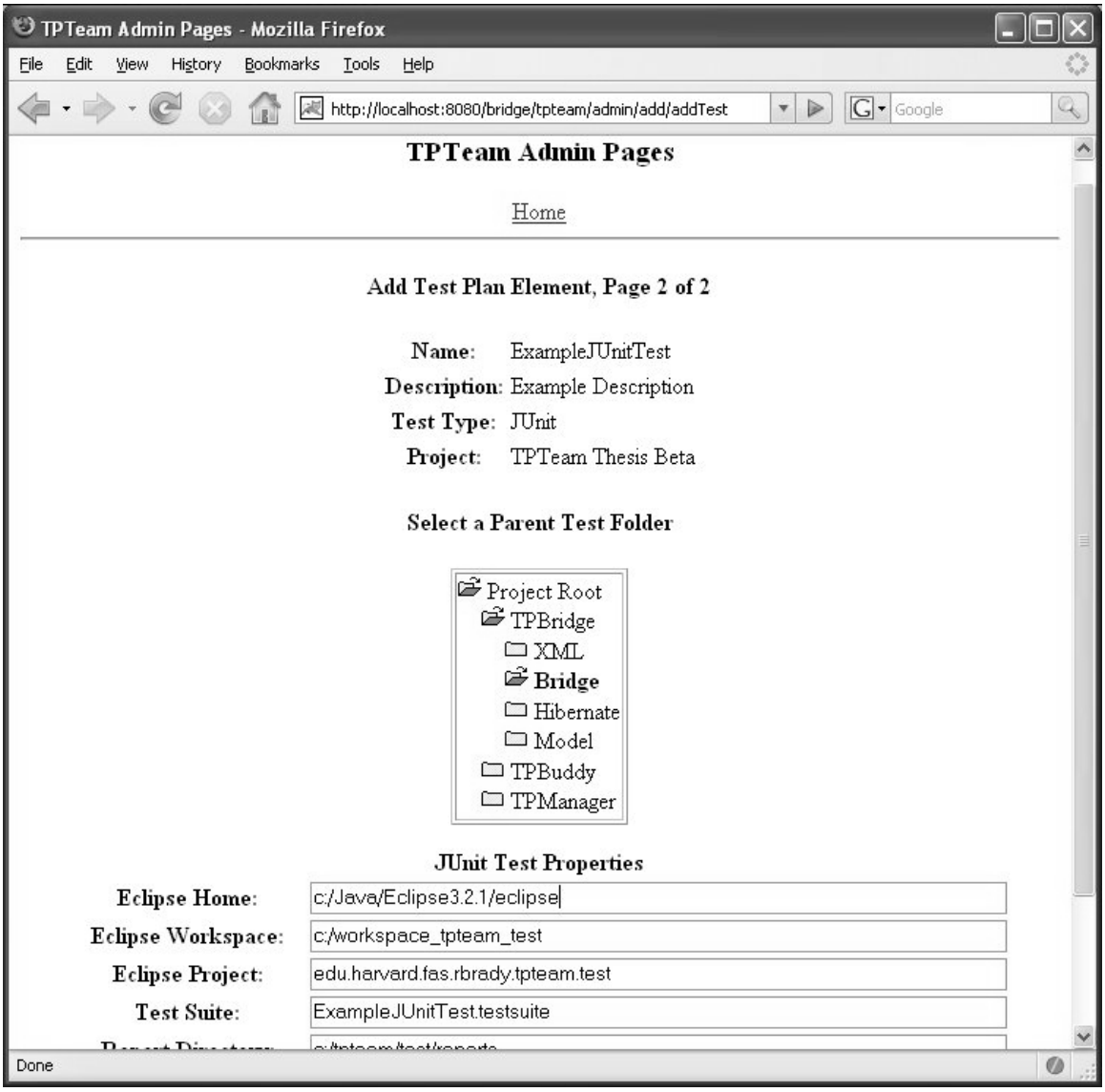

Figure 36 TPManager Add Test Definition Webpage

The updating and viewing of tests is done with procedures similar to those described above for products, projects, and users. The only difference being the JavaScript tree widget is used in place of tabulated selection fields.

Execution of Test Definitions

A test definition is executed by first selecting the project that is its parent. Next, the interface presents the JavaScript test tree. The user can than highlight the test definition to be executed and press execute button to request TPManager to begin the test run. Figure 37 below shows the webpage in action.

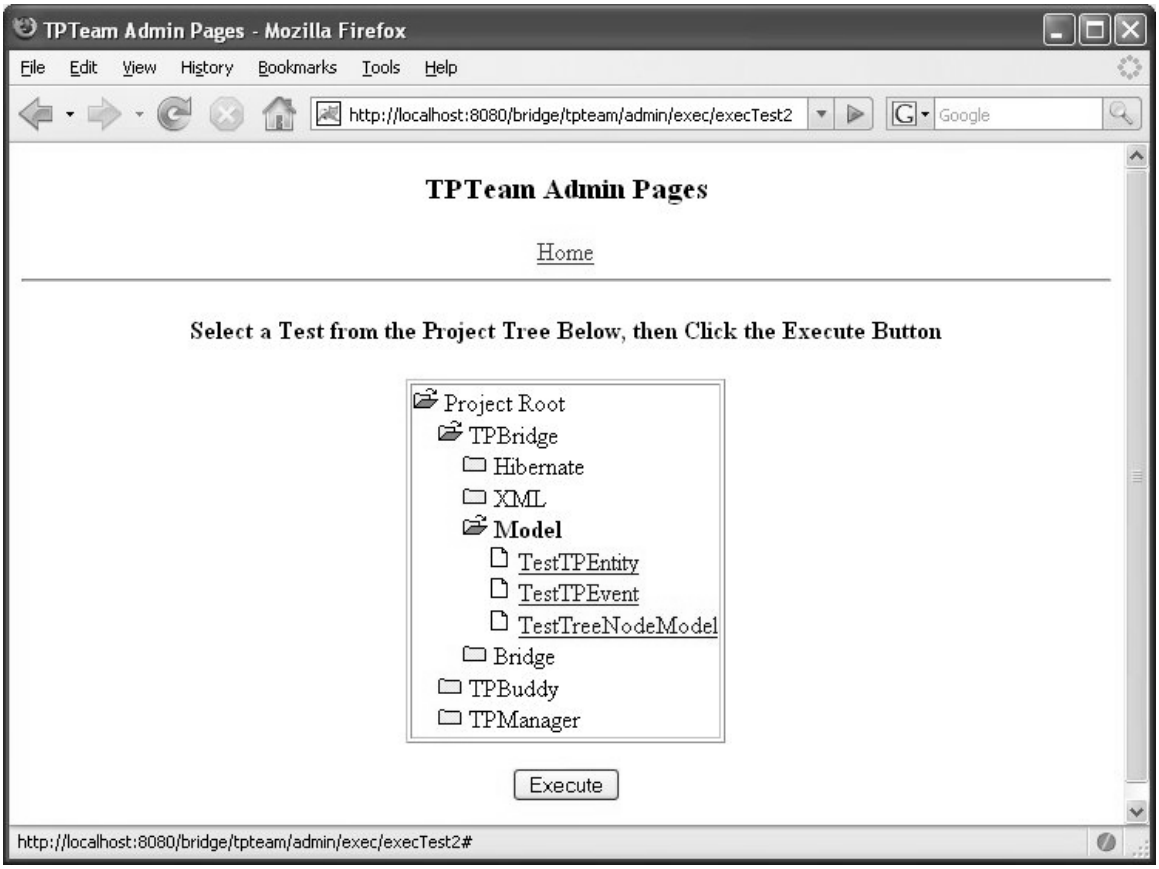

Figure 37 TPManager Test Execution Request Webpage

After TPManager has brokered the test execution and collected the results, the Web client's browser will be forwarded to a verdict page. An example is given in the following Figure 38.

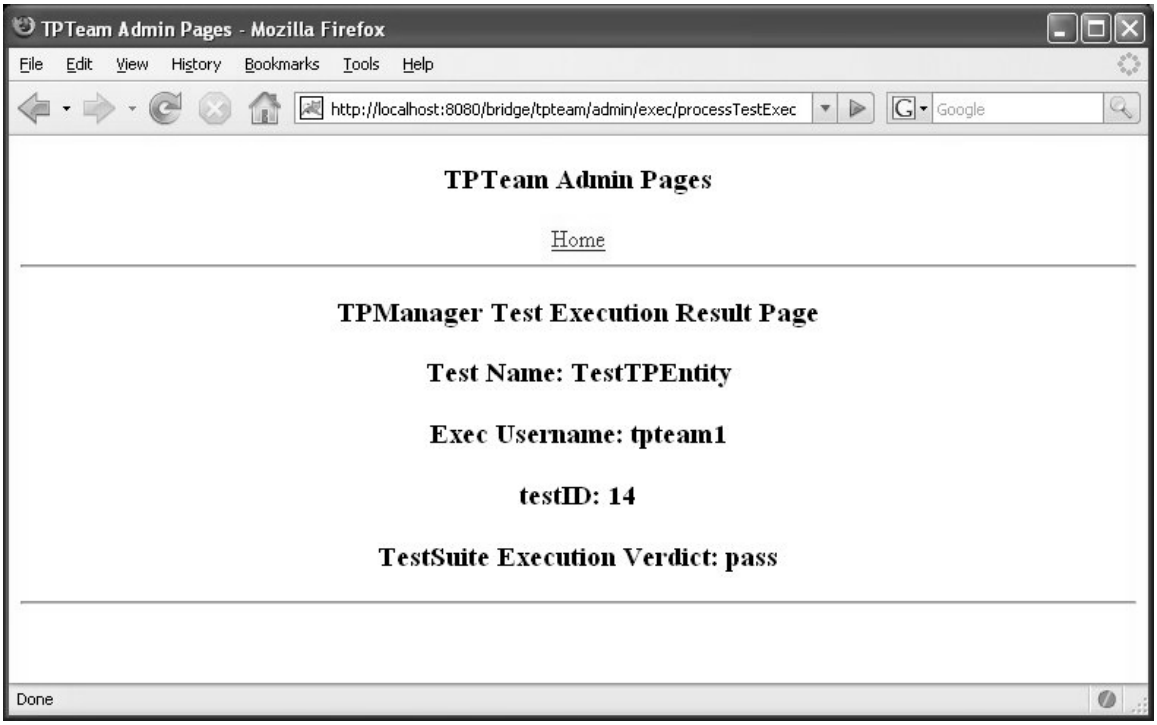

Figure 38 TPManager Test Execution Results Webpage

Uninstalling TPTeam

Uninstalling TPTeam is an easy task. Simply delete all unpacked zip directories that were created during the installation process. No Windows registry keys or other hidden configurations need be cleaned.

Chapter 5 Summary and Conclusions

This section discusses summarizes some of the software engineering lessons learned from the design and implementation of TPTeam. It concludes by pointing out areas of future work that may be of interest to developers looking to extend or utilized parts of TPTeam.

Lessons Learned

TPTeam utilizes and integrates many open source frameworks. Some of these platforms are mature and stable. Others are still in the incubation phase. TPTeam also uses several software components in different functional areas: database, serialization, instant messaging, etc. All of these combined to make a fertile, albeit sometimes frustrating, learning ground. The lessons learned will be summarized here.

Hibernate ORM Saves Time

The author was at first skeptical about introducing yet another framework into the project. However, it was found that the Hibernate ORM tool saved time during coding and deployment. One reason for the time savings was the elimination of tedious JDBC code that is normally written. Also, the Hibernate ant task that converts an object model to relational DDL statements was used often. This Hibernate feature allowed the author to: eliminate the hand-coding of schema creating SQL, keep the object model consistent with the relational one, and quickly change from the Oracle XE to MySQL database.

Distributed Systems: Serialization Matters

TPTeam is a distributed, event-driven system. The events in this case contain serialized TPEvent objects. The serialization implementation chosen has a significant impact on message latency and overall system response from a user perspective.

Based upon the author's commercial experience, a typical test project can have thousands of tests associated with it. Transmitting just the stubbed test tree GUI version of a thousand tests can take a substantial amount of time. For example, the one-way transmittal of a thousand stubbed tests was calculated to be 0.6s if the Sun Java 1.6 SDK built-in XML serialization was used. Transmission time calculations were performed for a custom serialization designed by the author and Betwixt also. The results are presented in Figure 39 below.

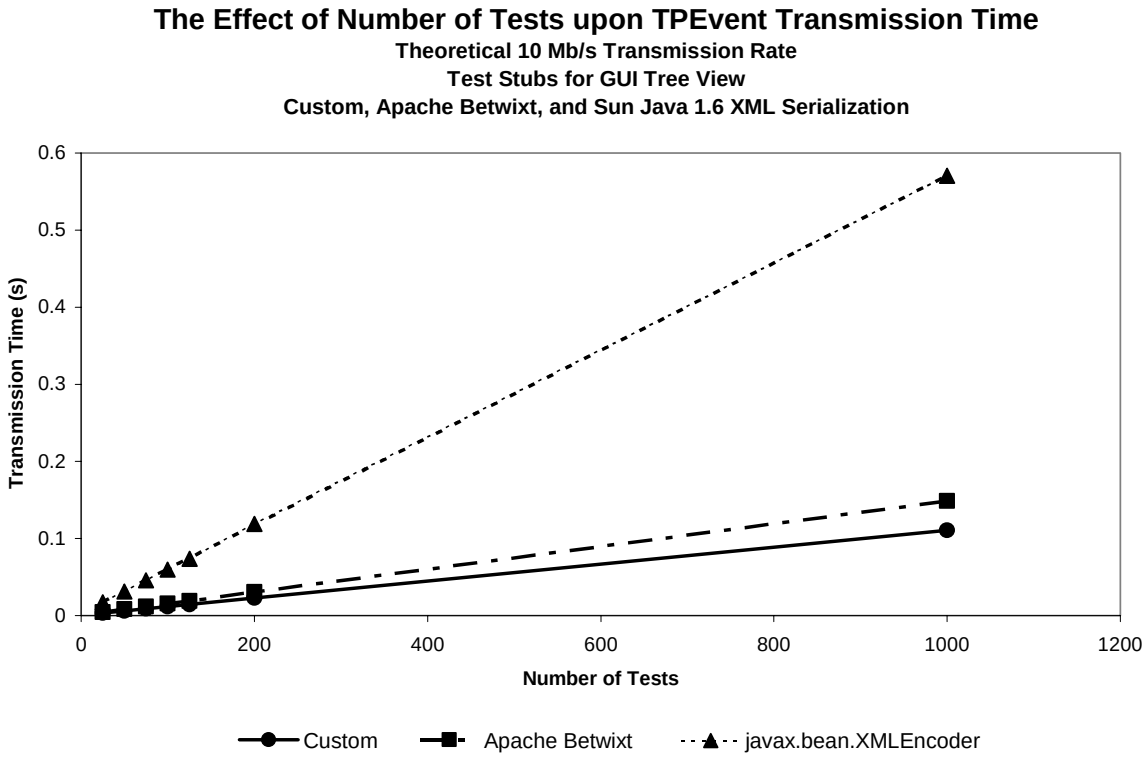

Figure 39 Theoretical Test Stub Transmission Times

As can be seen from Figure 39, the latency penalty imposed by using the standard Java JDK XM serialization is excessive. The one-way transmission for two-thousand test stubs would be over 1.2s compared to approximately 0.2s for the other serialization methods.

Furthermore, it was found that the small benefit derived from using a custom XML serialization was outweighed by the advantages of a well-documented library such as Betwixt. The advantages found were: order of magnitude less development effort, easier maintenance of code base, and built-in customization features present.

Open Source: Pros and Cons

TPTeam's use of free and open software has the obvious advantages of no license fees and code available for extension. However, there are many hidden costs with utilizing the open source too. For example, the ECF project had API-breaking releases nearly every two weeks during the course of TPTeam's development. The refactoring and debugging caused by these releases has been estimated by the author to be about ten to fifteen percent of the overall TPTeam implementation effort. Another case of hidden penalties is the TPTP project. Here, many features of remote agent JUnit and JUnit Plug-in testing are exposed through the API, but one later finds out that they are not implemented or are part of an IBM commercial product that is available only for a fee.

Future Work

TPTeam is also a fertile ground for areas of future work. Recommendations are summarized below.

- Addition of another open source ALM tool to TPTeam. For example, the open source Bugzilla (The Mozilla Foundation, 2007) bug tracking tool could be incorporated into the framework. TPEvents and the TPBridge would be extended to accommodate Bugzilla. Test failures could automatically trigger the assignment or triage of bugs based upon the test history and new algorithms embedded within TPTeam.
- Generalization of TPBridge. A more general purpose out-of-JVM communications bridge plug-in for Eclipse could be built from the TPBridge component. This new plug-in might have its own generic event object and a well-defined XML schema for its serialized messages.
- Extension of TPManager. Additional test types could be added: manual and Web performance, for example. The entire component could be refactored to run as a pure Web application instead of as an embedded Eclipse instance within a servlet container.

References

Borland Software Corp.(2007). JBuilder 2006 Datasheet, http://www.borland.com/resources/en/pdf/products/jbuilder/jb2006_datasheet.pdf, retrieved June 2007.

Bauer, C., and King, G. (2005). Hibernate in Action. Manning Publications, New York, 15.

Brady, B. (2006). EclipseWorld 2006 Presentation, http://tpteam.eclipse.googlepages.com/TPTeamEclipseWorld.ppt, retrieved June 2007.

CollabNet (2007). The Subversion Product Page, http://www.collab.net/products/open_source_subversion/, retrieved June 2007.

Cook, C., Irwin, W., and Churcher, N. (2005). A User Evaluation of Synchronous Collaborative Software Engineering Tools. 12th Asia-Pacific Software Engineering Conference (APSEC'05), 705-710.

Donsez, D. (2007). OSGi-JMS Bridge: http://www-adele.imag.fr/~donsez/dev/osgi/eajmsbridge/osgi-eajmsbridge.pdf, retrieved June 2007.

Gilbert, D. (2007). JFreeChart, http://www.jfree.org/jfreechart/, retrieved June 2007.

Google Inc. (2007). The Google Project Hosting Website, http://code.google.com/hosting/, retrieved June 2007.

OASIS (2007a). The Organization for the Advancement of Structured Information Web Services Distributed Management Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm, retrieved June 2007.

OASIS (2007b). The Organization for the Advancement of Structured Information Standards Service-Oriented Reference Model: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, retrieved June 2007.

OSGi (2007a). The Open Services Gateway Initiative, http://www.osgi.org/, retrieved June 2007.

OSGi (2007b). The Open Services Gateway Initiative Service Platform Programming Interface, http://bundles.osgi.org/Javadoc/r4/index.html, retrieved June 2007.

OSGi (2007c). The Open Services Gateway Initiative Service Platform Event Admin Service Specification, http://www.osgi.org/osgi_technology/download_specs.asp?section=2#Release4, retrieved June 2007.

Peh, D., Hannemann, A., and Hague, N. (2006). BIRT: A Field Guide to Reporting, Addison-Wesley Professional, Boston

Powell, G. (2005). Beginning Database Design. Wilely Publishing, Inc., Indianapolis, 73-122.

Red Hat (2007). The JBoss Hibernate datasheet, http://www.jboss.com/pdf/HibernateBrochure-03_07.pdf, retrieved June 2007.

Sourceforge Inc. (2007). The Sourceforge.net Open Source Software Development Website, http://sourceforge.net/, retrieved June 2007.

Sun Microsystems (2007a). JSE Datasheet, http://www.sun.com/software/Javaenterprisesystem/datasheet.pdf, retrieved June 2007.

Sun Microsystems (2007b). The JavaBeans 1.01 Specification, http://java.sun.com/products/javabeans/docs/spec.html, retrieved June 2007.

The Apache Software Foundation (2007). The Jakarta Commons Betwixt Project, http://jakarta.apache.org/commons/betwixt/, retrieved June 2007.

The Eclipse Foundation (2007a). The Eclipse Foundation Communication Framework, http://www.eclipse.org/ecf/, retrieved June 2007.

The Eclipse Foundation (2007b). The Eclipse Foundation Corona Project, http://www.eclipse.org/corona/, retrieved June 2007.

The Eclipse Foundation (2007c). Eclipse Platform Technical Overview, http://www.eclipse.org/whitepapers/eclipse-overview.pdf, retrieved June 2007.

The Eclipse Foundation (2007d). The Eclipse Foundation Business Intelligence and Reporting Tools Project, http://www.eclipse.org/birt/phoenix/, retrieved June 2007.

The Eclipse Foundation (2007e). The Eclipse Foundation Web Tools Platform, http://www.eclipse.org/webtools/main.php, retrieved June 2007.

The Eclipse Foundation (2007f). The Eclipse Foundation Application Lifecycle Framework Project, http://www.eclipse.org/alf/, retrieved June 2007.

The Eclipse Foundation (2007g). The Eclipse Foundation Equinox Project, http://www.eclipse.org/equinox/, retrieved June 2007.

The Eclipse Foundation (2007h). The Eclipse Foundation Test and Performance Tools Platform, http://www.eclipse.org/tptp/, retrieved June 2007.

The Eclipse Foundation (2007i). The Eclipse Foundation Rich Client Platform, http://wiki.eclipse.org/index.php/Rich_Client_Platform, retrieved June 2007.

The Eclipse Foundation (2007j). JPMorgan Raises the Bar for Banking Applications Case Study, http://www.eclipse.org/community/casestudies/jp_morgan_final.pdf, retrieved June 2007.

The Mozilla Foundation (2007). The Mozilla Foundation Bugzilla Project, http://www.bugzilla.org/, retrieved June 2007.

XMPP Standards Foundation (2007). The Extensible Messaging and Presence Protocol Specifications, http://www.xmpp.org/specs/, accessed June 2007.

Weathersby, J., French, D., Bondur, T., Tatchell, J., and Chatalbasheva, I. (2006). Integrating and Extending BIRT, Addison-Wesley Professional, Boston

Zeichick, A. (2006). Java Wars: Enterprise Developers Show Loyalty. Software Development Times, New York, 143, 5-12.

Appendix 1 Application Code

Appendix 1 contains listings of the TPTeam source code. The types of files listed in this Appendix and their order of presentation are given below.

- Java Code: The source for the Eclipse plug-ins.
- JavaScript Code: The source for the validation and confirmation functions used in TPManager's servlets.
- HTML Files: The static Web pages exposed by TPManager.
- Configuration Files: The Betwixt files used for XML serialization, Cascading Style Sheets (CSS) for the formatting of Web pages, and The TPTeam Properties file for the setting of the XMPP connection parameters.
- Database Scripts: The SQL scripts used to create and seed the TPTeam database.

Java Code

This section lists the Java source code files of TPTeam. The files listed by package membership. Each package and the files within it are list in alphabetical order by name.

package edu.harvard.fas.rbrady.tpteam.tpbridge

Activator.java

```
/*********************************************************************
 *
 * File     :     Activator.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Controls the lifecycle of the TPBridge Plug-in
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import org.eclipse.core.runtime.FileLocator;
import org.eclipse.core.runtime.Path;
import org.eclipse.core.runtime.Platform;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.TPBridge;
import
edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin.EventAdminClient;
import
edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin.EventAdminHandler;

/*********************************************************************
 * File           :     Activator.java
 *
 * Description    :     Controls the lifecycle of the TPBridge Plug-in
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
 * Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class Activator implements BundleActivator {

      /** The TPBridge OSGi Service POJO */
      private static TPBridge mTPBridge;
      /** The EventAdmin Service Event Handler */
      private static EventAdminHandler mEventAdminHandler;
      /** The Client of the EventAdmin Service */
      private static EventAdminClient mEventAdminClient;
      /** Directory where TPTeam Java Properties file is stored */
      public static final String TPTEAM_PROP_DIR = "data";
      /** File name of TPTeam Java Properties file */
      public static String TPTEAM_PROP_FILE = "tpteam.properties";
      /** TPTeam Property key=value pairs */
      private static Properties mTPTeamProps;

      /*
```

```
 * OSGi framework bundle start hook
 */
public void start(BundleContext context) throws Exception {
      loadTPTeamProps(context);
      mEventAdminHandler = new EventAdminHandler(context);
      mEventAdminClient = new EventAdminClient(context);
      mTPBridge = new TPBridge(context);
      Platform.getBundle("org.eclipse.equinox.event").start();
}

/*
 * OSGi framework bundle stop hook
 */
public void stop(BundleContext context) throws Exception {
      mTPBridge.stop(context);
}

/**
 * Getter
 * @return The EventAdmin Service Handler
 */
public static EventAdminHandler getEventAdminHandler()
{
      return mEventAdminHandler;
}

/**
 * Getter
 * @return The TPBridge Service POJO
 */
public static TPBridge getTPBridge()
{
      return mTPBridge;
}

/**
 * Getter
 * @return The EventAdmin Service client
 */
public static EventAdminClient getEventAdminClient()
{
      return mEventAdminClient;
}

/**
 * Getter
 * @return The TPTeam Properties
 */
public static Properties getTPTeamProps()
{
      return mTPTeamProps;
}

/**
 * Reads in the TPTeam property key=value pairs
 * @param context The bundle context
 */
```

```
	private void loadTPTeamProps(BundleContext context) {
		try {
			Platform.getLocation().toFile();
			InputStream is =
FileLocator.openStream(context.getBundle(),
				new Path(TPTEAM_PROP_DIR + "/" +
TPTEAM_PROP_FILE), false);
			mTPTeamProps = new Properties();
			mTPTeamProps.load(is);
			is.close();

		} catch (IOException ioe) {
			ioe.printStackTrace();
		}
	}

}
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge

Client.java

```
/**********************************************************************
 *
 * File     :     Client.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Convenience client for the TPBridge OSGi Service
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge;

import java.util.ArrayList;
import java.util.Properties;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.util.ECFException;
import org.hibernate.SessionFactory;
import org.osgi.framework.BundleContext;
import org.osgi.util.tracker.ServiceTracker;
import edu.harvard.fas.rbrady.tpteam.tpbridge.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/**********************************************************************
 * File           :     Client.java
 *
 * Description    :     Convenience client for the TPBridge OSGi
 *                      Service
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class Client {
      /** The ServiceTracker for the TPBridge Service */
      protected ServiceTracker mServiceTracker;

      /** The TPManager ECFID property key */
      public static final String TPMANAGER_ECFID_KEY =
"tpmanager.ecfID";

      /** The TPManager ECFID value */
      private String mTPMgrECFID = null;

      /** The TPManager ECF account password */
      public static final String TPMANAGER_ECFID_PASSWORD =
"tpmanager.password";

      /**
       * Default Constructor
       */
      public Client() {
```

```
	}

	/**
	 * Constructor
	 *
	 * @param context
	 *            The TPBridge Plug-in Context
	 */
	public Client(BundleContext context) {
		mServiceTracker = new ServiceTracker(context,
				ITPBridge.class.getName(), null);
		mServiceTracker.open();
	}

	/**
	 * Setter
	 *
	 * @param tpMgrECFID
	 *            The TPManager ECFID
	 */
	public void setTPMgrECFID(String tpMgrECFID) {
		mTPMgrECFID = tpMgrECFID;
	}

	/**
	 * Getter
	 *
	 * @return The TPManager ECFID
	 */
	public String getTPMgrECFID() {
		return mTPMgrECFID;
	}

	/**
	 * Setter
	 *
	 * @param serviceTracker
	 *            The TPBridge OSGi ServiceTracker
	 */
	public void setServiceTracker(ServiceTracker serviceTracker) {
		mServiceTracker = serviceTracker;
	}

	/**
	 * Getter
	 *
	 * @return The TPBridge OSGi ServiceTracker
	 */
	public ServiceTracker getServiceTracker() {
		return mServiceTracker;
	}

	/**
	 * Associates an ECF Communications Container with the TPBridge
Service
	 *
```

```
       * @param container
       *            The ECF Container
       * @param targetIDName
       *            The ECFID of the client
       * @param clientType
       *            The type of client
       * @return true if operation successful, false otherwise
       * @throws ECFException
       */
      public boolean setContainer(IContainer container, String
targetIDName,
                  String clientType) throws ECFException {
            ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
            if (tpBridge == null) {
                  System.out.println("TPBridgeClient: tpBridge service
ref is null");
                  return false;
            }
            tpBridge.setContainer(container, targetIDName, clientType);
            System.out
                        .println("TPBridgeClient: tpBridge service
container was set");
            return true;
      }

      /**
       * Getter
       *
       * @return The Hibernate database session factory
       * @throws RuntimeException
       */
      public SessionFactory getHibernateSessionFactory() throws
RuntimeException {
            ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
            if (tpBridge == null) {
                  System.out.println("TPBridgeClient: tpBridge service
ref is null");
                  throw new RuntimeException(

      "TPBridge.Client.getHibernateSessionFactory() Error: tpBridge
service ref is null.");
            }
            return tpBridge.getHibernateSessionFactory();
      }

      /**
       * Gets the list of TPTeam Events logged by the TPBridge OSGi
Service
       *
       * @return list log of TPTeam Events
       */
      public ArrayList<TPEvent> getEventLog() {
            ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
            if (tpBridge == null) {
```

```
                  System.out.println("TPBridgeClient: tpBridge service
ref is null");
            }
            return tpBridge.getEventLog();
      }

      /**
       * Getter
       *
       * @return the key=value properties associated with the TPBridge
       */
      public Properties getTPTeamProps() {
            return Activator.getTPTeamProps();
      }

      /**
       * Gets the ECFID of the client to the TPBridge Service
       *
       * @return the client ECFID
       */
      public String getTargetIDName() {
            ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
            if (tpBridge == null) {
                  System.out.println("TPBridgeClient: tpBridge service
ref is null");
            }
            return tpBridge.getTargetIDName();
      }

      /**
       * Determines if TPTeam shared object has been instantiated and
associated
       * with an ECF communications container
       *
       * @return true if TPSharedObject instantiated, false otherwise
       */
      public boolean isSharedObjectActive() {
            ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
            if (tpBridge == null) {
                  return false;
            }
            return tpBridge.isSharedObjectActive();

      }

}
```

ITBridge.java

```
/*********************************************************************
 *
 * File     :     ITPBridge.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Interface for the TPBridge OSGi Service
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge;

import java.util.ArrayList;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.util.ECFException;
import org.hibernate.SessionFactory;
import org.osgi.service.event.Event;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*********************************************************************
 * File           :     ITPBridge.java
 *
 * Description    :     Interface for the TPBridge OSGi Service
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *********************************************************************/
public interface ITPBridge {
      // Convenience public String variables used throughout TPTeam

      // TPTeam Event Topics
      public static final String TEST_EXEC_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testexecreq";

      public static final String TEST_EXEC_RESULT_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testexecresult";

      public static final String TEST_DETAIL_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testdetailreq";

      public static final String TEST_DETAIL_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testdetailresp";

      public static final String TEST_ADD_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testaddreq";

      public static final String TEST_ADD_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testaddresp";

      public static final String TEST_UPDATE_DATA_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testupdatedatreq";
```

```
      public static final String TEST_UPDATE_DATA_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testupdatedatresp";

      public static final String TEST_UPDATE_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testupdatereq";

      public static final String TEST_UPDATE_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testupdateresp";

      public static final String TEST_DEL_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testdelreq";

      public static final String TEST_DEL_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testdelresp";

      public static final String TEST_TREE_GET_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testtreegetreq";

      public static final String TEST_TREE_GET_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/testtreegetresp";

      public static final String PROJ_GET_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/projgetreq";

      public static final String PROJ_GET_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/projgetresp";

      public static final String CHART_GET_DATA_REQ_TOPIC =
"edu/harvard/fas/rbrady/tpteam/chartgetdatareq";

      public static final String CHART_GET_DATA_RESP_TOPIC =
"edu/harvard/fas/rbrady/tpteam/chartgetdataresp";

      // Metadata about TPTeam
      public static final String IMPLEMENTATION_TYPE =
"IMPLEMENTATION_TYPE";

      public static final String DEMO_IMPLEMENTATION_TYPE = "DEMO";

      public static final String CONTAINER_ID_KEY = "CONTAINER_ID";

      public static final String SHARED_OBJECT_ID_KEY =
"SHARED_OBJECT_ID";

      public static final String TARGET_ID_KEY = "TARGET_ID_KEY";

      public static final String DEFAULT_CONTAINER_TYPE =
"ecf.xmpp.smack";

      public static final String DEFAULT_SHARED_OBJECT_ID = "myobject";

      public static final String CLIENT_TYPE = "CLIENT_TYPE";

      public static final String TPTEAM_MGR = "TPTEAM_MGR";

      public static final String TPTEAM_BUDDY = "TPTEAM_BUDDY";
```

```
	public static final String BRIDGE_EA_CLIENT_TOPICS_KEY =
"tpbridge.eventadminclient.topics";

	/**
	 * Create a valid ECFID
	 *
	 * @param name
	 *            The String name of the ID
	 * @return The ECFID
	 */
	public ID createID(String name);

	/**
	 * Associates an ECF Communications Container with the TPBridge
Service
	 *
	 * @param container
	 *            The ECF Container
	 * @param targetIDName
	 *            The ECFID of the client
	 * @param clientType
	 *            The type of client
	 * @return true if operation successful, false otherwise
	 * @throws ECFException
	 */
	public void setContainer(IContainer container, String
targetIDName,
			String clientType) throws ECFException;

	/**
	 * Gets the ECFID of the client to the TPBridge Service
	 *
	 * @return the client ECFID
	 */
	public String getTargetIDName();

	/**
	 * Getter
	 *
	 * @return The client type
	 */
	public String getClientType();

	/**
	 * Getter
	 *
	 * @return The Hibernate database session factory
	 * @throws RuntimeException
	 */
	public SessionFactory getHibernateSessionFactory();

	/**
	 * Sends an OSGi Event to the OSGi EventAdmin Service
	 *
	 * @param event
	 *            the OSGi Event to be sent
	 * @return true if message sent successfully, false otherwise
```

```
         */
        public boolean sendECFTPMsg(Event event);

        /**
         * Gets the list of TPTeam Events logged by the TPBridge OSGi
Service
         *
         * @return list log of TPTeam Events
         */
        public ArrayList<TPEvent> getEventLog();

        /**
         * Determines if TPTeam shared object has been instantiated and
associated
         * with an ECF communications container
         *
         * @return true if TPSharedObject instantiated, false otherwise
         */
        public boolean isSharedObjectActive();
}
```

TPBridge.java

```
/*****************************************************************
 *
 * File     :     TPBridge.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Implementation of the ITPBridge OSGi Service
 *                Interface
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge;

import java.util.ArrayList;
import java.util.Hashtable;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.identity.IDCreateException;
import org.eclipse.ecf.core.identity.IDFactory;
import org.eclipse.ecf.core.sharedobject.ISharedObjectContainer;
import org.eclipse.ecf.core.util.ECFException;
import org.hibernate.SessionFactory;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventAdmin;
import org.osgi.util.tracker.ServiceTracker;
import edu.harvard.fas.rbrady.tpteam.tpbridge.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*****************************************************************
 * File           :     TPBridge.java
 *
 * Description    :     Implementation of the ITPBridge OSGi Service
 *                      Interface
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TPBridge implements ITPBridge {
      /** Map of the TPTeam key=value properties */
      private Hashtable<String, String> mTPBridgeProps = new
Hashtable<String, String>();
      /** The ServiceTracker for the EventAdmin Service */
      private ServiceTracker mServiceTracker;
      /** The TPTeam shared object instance */
      private TPSharedObject mSharedObject = null;
      /** The ECF communications container */
      private IContainer mContainer;
      /** The ECFID of the associated TPBridge Service client */
      private String mTargetIDName;
      /** The type of the TPBridge client */
      private String mClientType;
```

```
	/**
	 * Constructor
	 * @param conext The TPBridge plug-in context
	 */
	public TPBridge(BundleContext context) {
		mTPBridgeProps.put(IMPLEMENTATION_TYPE,
DEMO_IMPLEMENTATION_TYPE);
		start(context);
	}

	/**
	 * Gets the ECFID of the client to the TPBridge Service
	 *
	 * @return the client ECFID
	 */
	public String getTargetIDName() {
		return mTargetIDName;
	}

	/**
	 * Getter
	 *
	 * @return The client type
	 */
	public String getClientType() {
		return mClientType;
	}

	/**
	 * Gets an EventAdmin ServiceTracker when TPBridge plug-in
	 * starts its lifecycle
	 *
	 * @param context The TPBridge plug-in context
	 */
	public void start(BundleContext context) {

		context
				.registerService(ITPBridge.class.getName(),
this,
					mTPBridgeProps);

		mServiceTracker = new ServiceTracker(context,
EventAdmin.class
				.getName(), null);
		mServiceTracker.open();
	}

	/**
	 * Associates an ECF Communications Container with the TPBridge
Service
	 *
	 * @param container
	 *		The ECF Container
	 * @param targetIDName
	 *		The ECFID of the client
	 * @param clientType
	 *		The type of client
```

```
	 * @return true if operation successful, false otherwise
	 * @throws ECFException
	 */
	public synchronized void setContainer(IContainer container,
			String targetIDName, String clientType) throws
ECFException {
		mContainer = container;
		mTargetIDName = targetIDName;
		mClientType = clientType;
		createTrivialSharedObjectForContainer();
	}


	public boolean sendECFTPMsg(Event event) {
		try {
			TPEvent tpEvent = new TPEvent(event);
			String sendTo =
tpEvent.getDictionary().get(TPEvent.SEND_TO);

			if (sendTo == null ||
sendTo.trim().equalsIgnoreCase(""))
				return false;

			String[] ecfIDs = sendTo.split("/");
			for (String ecfID : ecfIDs) {
				if (ecfID.equalsIgnoreCase(mTargetIDName))
					continue;
				mSharedObject.getContext()
						.sendMessage(createID(ecfID),
tpEvent);
				System.out.println("TPBridge.sendECFTPMsg: sent
event "
						+ tpEvent.getTopic() + " to " +
ecfID);
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
		return true;
	}

	/**
	 * Gets the list of TPTeam Events logged by the TPBridge OSGi
Service
	 *
	 * @return list log of TPTeam Events
	 */
	public ArrayList<TPEvent> getEventLog() {
		return Activator.getEventAdminHandler().getEventLog();
	}

	/**
	 * Getter
	 *
	 * @return The Hibernate database session factory
	 * @throws RuntimeException
	 */
```

```
	public SessionFactory getHibernateSessionFactory() {
		return HibernateUtil.getSessionFactory();
	}

	/**
	 * Closes the EventAdmin ServiceTracker just before the TPBridge
	 * plug-in ends its lifecycle
	 *
	 * @param context The plug-in context
	 */
	public void stop(BundleContext context) {
		mServiceTracker.close();
	}

	/**
	 * Instantiates the TPSharedObject associated with the TPBridge
	 * Service, adds the EventAdmin Service handler as an observer,
	 * and associates itself with the ECF Container.
	 *
	 * @throws ECFException
	 */
	protected void createTrivialSharedObjectForContainer() throws
ECFException {
		// Create a new GUID for new TrivialSharedObject instance
		ID newID = IDFactory.getDefault().createStringID(
				TPSharedObject.class.getName());

		mSharedObject = new TPSharedObject();

	mSharedObject.addObserver(Activator.getEventAdminHandler());
		// Add shared object to container
		((ISharedObjectContainer) mContainer
				.getAdapter(ISharedObjectContainer.class))

	.getSharedObjectManager().addSharedObject(newID, mSharedObject,
					null);

	}

	/**
	 * Create a valid ECFID
	 *
	 * @param name
	 *            The String name of the ID
	 * @return The ECFID
	 */
	public ID createID(String name) {
		try {
			return IDFactory.getDefault().createID(
					mContainer.getConnectNamespace(), name);
		} catch (IDCreateException e) {
			e.printStackTrace();
			return null;
		}
	}

	/**
```

```
	 * Determines if TPTeam shared object has been instantiated and
associated
	 * with an ECF communications container
	 *
	 * @return true if TPSharedObject instantiated, false otherwise
	 */
	public boolean isSharedObjectActive() {
		if (mSharedObject != null)
			return mSharedObject.getContext().isActive();
		else
			return false;
	}

}
```

TPSharedObject.java

```
/*****************************************************************
 *
 * File     :     TPSharedObject.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Implementation of the SharedObject for TPTeam
 *                Messaging
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.bridge;

import java.util.Observable;
import org.eclipse.ecf.core.events.IContainerConnectedEvent;
import org.eclipse.ecf.core.events.IContainerDisconnectedEvent;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.sharedobject.ISharedObject;
import org.eclipse.ecf.core.sharedobject.ISharedObjectConfig;
import org.eclipse.ecf.core.sharedobject.ISharedObjectContext;
import org.eclipse.ecf.core.sharedobject.SharedObjectInitException;
import
org.eclipse.ecf.core.sharedobject.events.ISharedObjectActivatedEvent;
import
org.eclipse.ecf.core.sharedobject.events.ISharedObjectDeactivatedEvent;
import
org.eclipse.ecf.core.sharedobject.events.ISharedObjectMessageEvent;
import org.eclipse.ecf.core.util.Event;

/*****************************************************************
 * File           :     TPSharedObject.java
 *
 * Description    :     Implementation of the SharedObject for TPTeam
 *                      Messaging
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TPSharedObject extends Observable implements ISharedObject
{

      /** The ECF Configuration of the shared object */
    ISharedObjectConfig config = null;

    /**
     * Default Constructor
     */
    public TPSharedObject() {
        super();
    }

    /**
     * Getter
     * @return The ECFID of the shared object
```

```
     */
    protected ID getID() {
        return config.getSharedObjectID();
    }

    /**
     * Getter
     * @return The ECF context of the shared object
     */
    protected ISharedObjectContext getContext() {
        if (config == null) return null;
        else return config.getContext();
    }

    /**
     * Initializes the shared object
     */
    public void init(ISharedObjectConfig initData)
            throws SharedObjectInitException {
        this.config = initData;
    }

    /**
     * Handles the all possible ECF Events that the shared object
     * may experience from its ECF communications container
     */
    public void handleEvent(Event event) {
      // First handle info type events
        if (event instanceof ISharedObjectActivatedEvent) {
            System.out.println("HELLO WORLD from "+getID()+".  I'm
activated!");
        } else if (event instanceof ISharedObjectDeactivatedEvent) {
            System.out.println("GOODBYE from "+getID()+".  I'm
deactivated!");
        } else if (event instanceof IContainerConnectedEvent) {
            System.out.println("Remote
"+((IContainerConnectedEvent)event).getTargetID()+" joined!");
        } else if (event instanceof IContainerDisconnectedEvent) {
            System.out.println("Remote
"+((IContainerDisconnectedEvent)event).getTargetID()+" departed!");
        // Now handle TPTeam Events that need to be acted upon by
observing objects
        } else if (event instanceof ISharedObjectMessageEvent) {
            setChanged();
            notifyObservers(event);
        }
    }

    /**
     * Handles many ECF Events at once
     */
    public void handleEvents(Event[] events) {
        for(int i=0; i < events.length; i++) {
            this.handleEvent(events[i]);
        }
    }
```

```
    /**
     * Dispose of this shared object
     */
    public void dispose(ID containerID) {
        this.config = null;
    }

    /**
     * Required ECF ISharedObject interface method, not used.
     */
    @SuppressWarnings("unchecked")
      public Object getAdapter(Class clazz) {
        return null;
    }
}
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.chart

ChartDataPoint.java

```
/*********************************************************************
 *
 * File     :     ChartDataPoint.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A serializable TPTeam test execution data point to be
 *                encapsulated within a TPTeam Event or ChartDataSet
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.chart;

import java.util.Date;

/*********************************************************************
 * File           :     ChartDataPoint.java
 *
 * Description    :     A serializable TPTeam test execution data point
 *                      to be encapsulated within a TPTeam Event or
 *                      ChartDataSet
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady
*********************************************************************/
public class ChartDataPoint implements java.io.Serializable {
      /** ID used during serialization */
      private static final long serialVersionUID = 1L;
      /** Date retrieved from database corresponding to data creation
timestamp */
      private Date mDate;
      /** Number of successful test executions */
      private int mPass;
      /** Number of test executions that ran, but failed */
      private int mFail;
      /** Number of test executions that did not run due to errors */
      private int mError;
      /** Number of test execution inconclusive due to connection time
out, etc. */
      private int mInconcl;
      /** Number of test definitions that were never requested to be
run */
      private int mNotExec;

      /**
       * Default constructor
       */
      public ChartDataPoint()
      {
```

```
}

/**
 * Setter
 * @param date the data creation time from database
 */
public void setDate(Date date) {
      mDate = date;
}

/**
 * Getter
 * @return the data creation time from database
 */
public Date getDate() {
      return mDate;
}

/**
 * Setter
 * @param pass the number of passed test executions
 */
public void setPass(int pass) {
      mPass = pass;
}

/**
 * Getter
 * @return the number of passed test executions
 */
public int getPass() {
      return mPass;
}

/**
 * Setter
 * @param fail the number of failed executions
 */
public void setFail(int fail) {
      mFail = fail;
}

/**
 * Getter
 * @return the number of failed test executions
 */
public int getFail() {
      return mFail;
}

/**
 * Setter
 * @param error the number of erroneous test executions
 */
public void setError(int error) {
      mError = error;
}
```

```
	/**
	 * Getter
	 * @return the number of erroneous test executions
	 */
	public int getError() {
		return mError;
	}

	/**
	 * Setter
	 * @param inconcl the number of inconclusive test executions
	 */
	public void setInconcl(int inconcl) {
		mInconcl = inconcl;
	}

	/**
	 * Getter
	 * @return the number of inconclusive test executions
	 */
	public int getInconcl() {
		return mInconcl;
	}

	/**
	 * Setter
	 * @param notExec the number of test executions never executed
	 */
	public void setNotExec(int notExec) {
		mNotExec = notExec;
	}

	/**
	 * Getter
	 * @return the number of test executions never executed
	 */
	public int getNotExec() {
		return mNotExec;
	}
}
```

ChartDataSet.java

```
/*********************************************************************
 *
 * File     :     ChartDataSet.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A serializable TPTeam test execution data set to be
 *                encapsulated within a TPTeam Event.
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.chart;

import java.util.ArrayList;
import java.util.List;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;

/*********************************************************************
 * File           :     ChartDataSet.java
 *
 * Description    :     A serializable TPTeam test execution data set
 *                      to be encapsulated within a TPTeam Event.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 366 $
 * @date $Date: 2007-06-14 22:38:58 -0400 (Thu, 14 Jun 2007) $
Copyright (c) 2007 Bob Brady

*********************************************************************/
public class ChartDataSet implements java.io.Serializable,
Comparable<ChartDataSet> {
      /** ID used during serialization */
      private static final long serialVersionUID = 1L;
      /** The TPTeam user who is associated with the data points of the
set */
      private TpteamUser mUser;
      /** The chart type associated with the data set: Pie, Bar, or
Line */
      private String mType;
      /** The name of the TPTeam project associated with the data set
*/
      private String mProjName;
    /** The list of data points of the set */
      private List<ChartDataPoint> mChartDataPoints = new
ArrayList<ChartDataPoint>();

      // Convenience public Strings representing chart types
      public static final String PIE = "PIE_CHART";
      public static final String BAR = "BAR_CHART";
      public static final String LINE = "LINE_CHART";
      public static final String CHART_TYPE = "CHART_TYPE";

      /**
       * Default Constructor
       */
      public ChartDataSet()
```

```
{

}

/**
 * Setter
 * @param user The TPTeam user
 */
public void setUser(TpteamUser user) {
      mUser = user;
}

/**
 * Getter
 * @return the TPTeam user
 */
public TpteamUser getUser() {
      return mUser;
}

/**
 * Setter
 * @param type The chart type
 */
public void setType(String type) {
      mType = type;
}

/**
 * Getter
 * @return the chart type
 */
public String getType() {
      return mType;
}

/**
 * Setter
 * @param projName the TPTeam project name
 */
public void setProjName(String projName)
{
      mProjName = projName;
}

/**
 * Getter
 * @return the TPTeam project name
 */
public String getProjName()
{
      return mProjName;
}

/**
 * Setter
 * @param chartDataPoints the data points of the data set
```

```
       */
      public void setChartDataPoints(List<ChartDataPoint>
chartDataPoints) {
            mChartDataPoints = chartDataPoints;
      }

      /**
       * Getter
       * @return the data points of the data set
       */
      public List<ChartDataPoint> getChartDataPoints() {
            return mChartDataPoints;
      }

      /**
       * Adds a data point to the list of data points
       * @param chartDataPoint the data point to add
       */
      public void addChartDataPoint(ChartDataPoint chartDataPoint) {
                  mChartDataPoints.add(chartDataPoint);
      }

      /**
       * Compares one data set to another.
       *
       *    Returns -1 if this data set TPTeam user is alphabetically
       * before parameter data set's TPTeam user, 0 if user names
       * are equal, and 1 if parameter data set's TPTeam user is
       * alphabetically before this data set's TPTeam user.
       *
       * @return  -1, 0, or 1
       */
      public int compareTo(ChartDataSet chartDataSet) {
            return this.mUser.compareTo(chartDataSet.getUser());
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin

EventAdminClient.java

```
/**********************************************************************
 *
 * File     :     EventAdminClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A client of the EventAdmin Service
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin;

import java.util.Hashtable;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventAdmin;
import org.osgi.util.tracker.ServiceTracker;

/**********************************************************************
 * File           :     EventAdminClient.java
 *
 * Description    :     A client of the EventAdmin Service.  It sends
 *                      TPTeam Event data encapsulated as an OSGi Event
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 371 $
 * @date $Date: 2007-06-16 17:56:02 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class EventAdminClient {
      /** The EventAdmin ServiceTracker */
      private ServiceTracker mServiceTracker;

      /**
       * Constructor, gets and opens a tracker for
       * the EventAdmin OSGi Service
       *
       * @param context The TPBridge plug-in context
       */
      public EventAdminClient(BundleContext context) {
            mServiceTracker = new ServiceTracker(context,
EventAdmin.class
                        .getName(), null);
            mServiceTracker.open();
      }

      /**
       * Sends an OSGi Event to the OSGi EventAdmin Service
       * @param topic the event topic String
       * @param dictionary a table of key=value pairs
       * @return true if message sent successfully, false otherwise
       */
```

```
      public boolean sendEvent(String topic, Hashtable<String, String>
dictionary) {
            boolean messageSent = false;

            EventAdmin eventAdmin = (EventAdmin)
mServiceTracker.getService();

            if (eventAdmin != null) {

                        System.out.println("TPBridge EventAdminClient:
Sent " + topic + " Event" );
                  eventAdmin.sendEvent(new Event(topic, dictionary));
                  messageSent = true;
            }
            return messageSent;
      }

}
```

EventAdminHandler.java

```
/*****************************************************************
 *
 * File     :     EventAdminHandler.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The OSGi Service that handles OSGi Events from the
 *                EventAdmin Service and acts as an observer of TPTeam
 *                Events received by the TPSharedObject of the TPBridge
 *                OSGi Service
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.eventadmin;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Observable;
import java.util.Observer;
import
org.eclipse.ecf.core.sharedobject.events.ISharedObjectMessageEvent;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventConstants;
import org.osgi.service.event.EventHandler;
import edu.harvard.fas.rbrady.tpteam.tpbridge.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.TPSharedObject;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*****************************************************************
 * File           :     EventAdminHandler.java
 *
 * Description    :     The OSGi Service that handles OSGi Events from
 *                      the EventAdmin Service and acts as an observer
 *                      of TPTeam Events received by the TPSharedObject
 *                      of the TPBridge OSGi Service
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 371 $
 * @date $Date: 2007-06-16 17:56:02 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class EventAdminHandler implements EventHandler, Observer {

      /** Table of key=value pairs of the OSGi Event */
      private Hashtable<String, String[]> mDictionary = new
Hashtable<String, String[]>();
      /** A List log of all TPEvents received */
      private ArrayList<TPEvent> mEvents;

      /**
       * Constructor
       *
       * Sets the OSGi Event topics to be handled by this service
       *
```

```
       * Sets the
       * @param context
       */
      public EventAdminHandler(BundleContext context) {
            mEvents = new ArrayList<TPEvent>();
            // Set topics for events where tpbridge will send out of
JVM on ECF
            // SharedObect
            String topicsProp = Activator.getTPTeamProps().getProperty(
                        ITPBridge.BRIDGE_EA_CLIENT_TOPICS_KEY);
            String topics[] = topicsProp.split(",");
            for (String topic : topics)

      System.out.println(ITPBridge.BRIDGE_EA_CLIENT_TOPICS_KEY + ": "
                              + topic);

            mDictionary.put(EventConstants.EVENT_TOPIC, topics);
            context
                        .registerService(EventHandler.class.getName(),
this,
                                    mDictionary);
      }

      /**
       * Handles an OSGi Event from the EventAdmin Service
       * and fowards it to the TPBridge Service so it can
       * be sent out-of-JVM
       *
       * @param event the OSGi Event to be handled
       */
      public void handleEvent(Event event) {
            // Info message
            System.out.println("TPBridge EventAdminHandler: Got "
                        + event.getTopic() + " Event SEND_TO "
                        + event.getProperty(TPEvent.SEND_TO));
            // Adapt the OSGi Event to a TPEvent and log it
            TPEvent tpEvent = new TPEvent(event);
            mEvents.add(tpEvent);

            // If no ECF recipient, skip sending
            String sendTo =
tpEvent.getDictionary().get(TPEvent.SEND_TO);
            if (sendTo == null || sendTo.trim().equalsIgnoreCase(""))
                  return;

            // Send out-of-JVM via TPBridge OSGi Service
            Activator.getTPBridge().sendECFTPMsg(event);
      }

      /**
       * Handles incoming out-of-JVM messages from the TPBridge
       * shared object and routes to all subscribers via the
       * OSGi EventAdmin Service
       *
       * @param observable the TPSharedObject of the TPBridge Service
       * @param object the ECF shared object message event to be
handled
```

```
	 */
	public void update(Observable observable, Object object) {
		if (observable instanceof TPSharedObject
				&& object instanceof ISharedObjectMessageEvent)
{
			// Extract the encapsulated TPEvent
			ISharedObjectMessageEvent sharedObjEvent =
(ISharedObjectMessageEvent) object;
			TPEvent tpEvent = (TPEvent) sharedObjEvent.getData();
			System.out
					.println("TPBridge: Update from
SharedObject Got TPEvent topic "
							+ tpEvent.getTopic() + " from
" + tpEvent.getDictionary().get(TPEvent.FROM));
			// Included so outbound messages don't get into
infinite loop
			if(isTopicForwardable(tpEvent.getTopic()))
			{

	Activator.getEventAdminClient().sendEvent(tpEvent.getTopic(),
					tpEvent.getDictionary());
			}
		}
	}

	/**
	 * Getter
	 * @return the log list of TPEvents
	 */
	public ArrayList<TPEvent> getEventLog() {
		return mEvents;
	}

	/**
	 * Helper method to determine if an event having the given
	 * topic should be forwarded out-of-JVM by the TPBridge
	 * OSGi Service
	 * @param topic the topic of an event
	 * @return true if it should be forwarded out-of-JVM, false
otherwise
	 */
	private boolean isTopicForwardable(String topic)
	{
		for(String bridgeTopic :
mDictionary.get(EventConstants.EVENT_TOPIC))
		{
			if(topic.equalsIgnoreCase(bridgeTopic))
				return false;
		}
		return true;
	}
}
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate

HibernateUtil.java

```
/*********************************************************************
 *
 * File     :     HibernateUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides Hibernate framework utility methods
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate;
import org.hibernate.*;
import org.hibernate.cfg.*;

/*********************************************************************
 * File           :     HibernateUtil.java
 *
 * Description    :     Provides Hibernate framework utility methods
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
 * Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class HibernateUtil {

      /** The Hibernate Database Connection Session Factory */
      private static final SessionFactory sessionFactory;

      static {
            try {
                  // Create the SessionFactory from hibernate.cfg.xml
                  sessionFactory = new Configuration().configure()
                              .buildSessionFactory();
            } catch (Throwable ex) {
                  // Make sure you log the exception, as it might be
swallowed
                  System.err.println("Initial SessionFactory creation
failed." + ex);
                  throw new ExceptionInInitializerError(ex);
            }
      }

      /**
       * Getter
       * @return The Hibernate Database Session Factory
       */
      public static SessionFactory getSessionFactory() {
            return sessionFactory;
      }
}
```

Test.java

```
/*****************************************************************
 *
 * File     :     Test.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Hibernate generated model class supplemented with
 *                      helper methods
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

/*****************************************************************
 * File           :     Test.java
 *
 * Description    :     Hibernate generated model class supplemented
 *                      with helper methods
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
 * Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class Test implements java.io.Serializable {

      // Fields

      private static final long serialVersionUID = 1L;

      private int id;

      private Project project;

      private TestType testType;

      private TpteamUser createdBy;

      private TpteamUser modifiedBy;

      private Test parent;

      private String name;

      private String description;

      private Character isFolder;

      private String path;

      private Date createdDate;
```

```
	private Date modifiedDate;

	private Set<TestExecution> testExecutions = new
HashSet<TestExecution>(0);

	private Set<Test> children = new HashSet<Test>(0);

	private Set<JunitTest> junitTests = new HashSet<JunitTest>(0);

	// Constructors

	/** default constructor */
	public Test() {
	}

	/** minimal constructor */
	public Test(int id) {
		this.id = id;
	}

	/** full constructor */
	public Test(int id, Project project, TestType testType,
			TpteamUser createdBy, Test parent, String name,
String description,
			Character isFolder, String path, Set<TestExecution>
testExecutions,
			Set<Test> children, Set<JunitTest> junitTests) {
		this.id = id;
		this.project = project;
		this.testType = testType;
		this.createdBy = createdBy;
		this.parent = parent;
		this.name = name;
		this.description = description;
		this.isFolder = isFolder;
		this.path = path;
		this.testExecutions = testExecutions;
		this.children = children;
		this.junitTests = junitTests;
	}

	// Property accessors
	public int getId() {
		return this.id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public Project getProject() {
		return this.project;
	}

	public void setProject(Project project) {
		this.project = project;
	}
```

```
public TestType getTestType() {
	return this.testType;
}

public void setTestType(TestType testType) {
	this.testType = testType;
}

public TpteamUser getCreatedBy() {
	return this.createdBy;
}

public void setCreatedBy(TpteamUser createdBy) {
	this.createdBy = createdBy;
}

public TpteamUser getModifiedBy() {
	return this.modifiedBy;
}

public void setModifiedBy(TpteamUser modifiedBy) {
	this.modifiedBy = modifiedBy;
}

public Test getParent() {
	return this.parent;
}

public void setParent(Test parent) {
	this.parent = parent;
}

public String getName() {
	return this.name;
}

public void setName(String name) {
	this.name = name;
}

public String getDescription() {
	return this.description;
}

public void setDescription(String description) {
	this.description = description;
}

public Character getIsFolder() {
	return this.isFolder;
}

public void setIsFolder(Character isFolder) {
	this.isFolder = isFolder;
}
```

```
	public String getPath() {
		return this.path;
	}

	public void setPath(String path) {
		this.path = path;
	}

	public Date getCreatedDate() {
		return this.createdDate;
	}

	public void setCreatedDate(Date createdDate) {
		this.createdDate = createdDate;
	}

	public Date getModifiedDate() {
		return this.modifiedDate;
	}

	public void setModifiedDate(Date modifiedDate) {
		this.modifiedDate = modifiedDate;
	}

	public Set<TestExecution> getTestExecutions() {
		return this.testExecutions;
	}

	public void setTestExecutions(Set<TestExecution> testExecutions)
{
		this.testExecutions = testExecutions;
	}

	public Set<Test> getChildren() {
		return this.children;
	}

	public void setChildren(Set<Test> children) {
		this.children = children;
	}

	public void addChild(Test child) {
		this.children.add(child);
	}

	public Set<JunitTest> getJunitTests() {
		return this.junitTests;
	}

	public void setJunitTests(Set<JunitTest> junitTests) {
		this.junitTests = junitTests;
	}

	public void addJunitTest(JunitTest junitTest)
	{
		this.junitTests.add(junitTest);
	}
```

```
	/**
	 * Utility method for printing the test node
	 * tree to a given depth starting with this
	 * node
	 *
	 * @param depth
	 */
	public void printNode(int depth) {
		String pad = "";
		for (int idx = 0; idx < depth; idx++)
			pad += "\t";
		System.out.println(pad + "printNode: " + name);
		for (Test child : getChildren())
			child.printNode(depth + 1);
	}

	/**
	 * Initializes a Test "Skeleton" for use in displaying Tests in
GUI trees
	 *
	 * Recursively initializes ID, Name, Description, isFolder,
testType,
	 * parent, for this Test and all its children
	 */
	public void initSkeleton() {
		getId();
		getName();
		getDescription();
		getIsFolder();
		Test parent = getParent();
		TestType testType = getTestType();
		if (parent != null)
			parent.getId();
		if (testType != null)
			testType.getName();
		for (Test child : getChildren())
			child.initSkeleton();
	}

	public void initProps(boolean includeExec) {
		getId();
		getName();
		getDescription();
		getCreatedBy().getFirstName();
		getCreatedBy().getLastName();
		getCreatedDate();
		getIsFolder();
		TestType testType = getTestType();
		if (testType != null)
			testType.getName();
		if (getModifiedBy() != null) {
			getModifiedBy().getFirstName();
			getModifiedBy().getLastName();
			getModifiedDate();
		}
		if (getJunitTests() != null) {
```

```
                        for (JunitTest junit : getJunitTests()) {
                              junit.getEclipseHome();
                              junit.getWorkspace();
                              junit.getProject();
                              junit.getTestSuite();
                              junit.getReportDir();
                              junit.getTptpConnection();
                        }
                  }
                  if (includeExec && getTestExecutions() != null) {
                        for (TestExecution testExec : getTestExecutions()) {
                              testExec.getStatus();
                              testExec.getTpteamUser().getFirstName();
                              testExec.getTpteamUser().getLastName();
                              testExec.getTpteamUser().getEcfId();
                              testExec.getExecDate();
                              testExec.getComments();
                        }
                  }
            }

      }
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.model

AbstractTreeNode.java

```
/*********************************************************************
 *
 * File     :     AbstractTreeNode.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Base class for tree nodes
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/*********************************************************************
 * File           :     AbstractTreeNode.java
 *
 * Description    :     Base class for tree nodes
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *********************************************************************/
public abstract class AbstractTreeNode implements ITreeNode {

      /** The TPTeam ID of the tree node */
      private String mID;
      /** The TPTeam name of the tree node */
      private String mName;
      /** The parent of this tree node */
      private ITreeNode mParent = null;
      /** List of all listeners of this tree node */
      private List<ITreeNodeChangeListener> mChangeListeners;
      /** The children of this tree node */
      private List<ITreeNode> mChildren = new ArrayList<ITreeNode>();

      // Property accessors

      public List<ITreeNode> getChildren() {
            return mChildren;
      }

      public void setChildren(List<ITreeNode> children) {
            for (ITreeNode child : children)
                  mChildren.add(child);
      }

      public boolean hasChildren() {
            if (mChildren == null || mChildren.size() < 1)
```

```
            return false;
      return true;
}

public void addChild(ITreeNode child) {
      mChildren.add(child);
      fireNodeAdded(child);
}

public boolean removeChild(ITreeNode child) {
      boolean isRemoved = mChildren.remove(child);
      if (isRemoved)
            fireNodeDeleted(child);
      return isRemoved;
}

public boolean removeChild(int childID) {
      for (ITreeNode childEnt : mChildren) {
            if (((TPEntity) childEnt).getID().equals(childID)) {
                  boolean isRemoved = removeChild(childEnt);
                  if (isRemoved) {
                        fireNodeDeleted(childEnt);
                        return isRemoved;
                  }
            }
      }
      return false;
}


public String getID() {
      return mID;
}

public void setID(String id) {
      mID = id;
}

public String getName() {
      return mName;
}

public void setName(String name) {
      mName = name;
      fireNodeUpdated();
}

public ITreeNode getParent() {
      return mParent;
}

public void setParent(ITreeNode parent) {
      mParent = parent;
}

public String toString() {
      return mName;
```

```
	}

	public void addChangeListener(ITreeNodeChangeListener listener) {
		if (mChangeListeners == null)
			mChangeListeners = new
ArrayList<ITreeNodeChangeListener>();

		/* if listener already exists, then do not add */
		if (mChangeListeners.contains(listener))
			return;

		mChangeListeners.add(listener);
	}

	public void removeChangeListener(ITreeNodeChangeListener
listener) {
		if (mChangeListeners == null)
			return;

		mChangeListeners.remove(listener);
	}

	protected List<ITreeNodeChangeListener> getChangedListeners() {
		if (mChangeListeners == null) {
			return Collections.emptyList();
		}
		return mChangeListeners;
	}

	/**
	 * Notifies all listeners when this node has had
	 * one of its properties updated
	 */
	protected void fireNodeUpdated() {
		for (ITreeNodeChangeListener listener :
getChangedListeners())
			listener.updateNode(this);
	}

	/**
	 * Notifies all listeners when this node adds
	 * a child node
	 *
	 * @param child the child tree node to be added
	 */
	protected void fireNodeAdded(ITreeNode child) {
		for (ITreeNodeChangeListener listener :
getChangedListeners())
			listener.addNode(child);
	}

	/**
	 * Notifies all listeners when this node deletes
	 * a child node
	 *
	 * @param child the child tree node to be deleted
	 */
```

```
      protected void fireNodeDeleted(ITreeNode child) {
            for (ITreeNodeChangeListener listener :
getChangedListeners()) {
                  System.out.println("Node " + this.getName() + "
fireNodeDeleted");
                  listener.deleteNode(child);
            }
      }

}
```

ITreeNode.java

```
/*********************************************************************
 *
 * File     :     ITreeNode.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The interface for TreeNode objects
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

import java.io.Serializable;
import java.util.List;

/*********************************************************************
 * File           :     ITreeNode.java
 *
 * Description    :     The interface for TreeNode objects`
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007)
 * $Copyright (c) 2007 Bob Brady
 *********************************************************************/
public interface ITreeNode extends Serializable {
      // Propert accessors
      public String getID();
      public void setID(String id);
      public String getName();
      public void setName(String name);
      public List<ITreeNode> getChildren();
      public void setChildren(List<ITreeNode> children);
      public void addChild(ITreeNode child);
      public boolean removeChild(ITreeNode child);
      public boolean removeChild(int childID);
      public boolean hasChildren();
      public ITreeNode getParent();
      public void setParent(ITreeNode parent);
      public String toString();
      public void addChangeListener(ITreeNodeChangeListener listener);
      public void removeChangeListener(ITreeNodeChangeListener
listener);
}
```

ITreeNodeChangeListener.java

```
/**********************************************************************
 *
 * File     :     ITreeNodeChangeListener.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The interface for listeners of ITreeNode
 *                implementations
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

/**********************************************************************
 * File           :     ITreeNodeChangeListener.java
 *
 * Description    :     The interface for listeners of ITreeNode
 *                      implementations
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 **********************************************************************/
public interface ITreeNodeChangeListener {

      /**
       * Performs actions when an ITreeNode
       * updates on of its properties
       *
       * @param node the ITreeNode that was updated
       */
      void updateNode(ITreeNode node);

      /**
       * Performs actions when an ITreeNode
       * adds a child node
       *
       * @param node the child node that was added
       */
      void addNode(ITreeNode node);

      /**
       * Performs actions when an ITreeNode
       * deletes a child node
       * @param node the child node that was deleted
       */
      void deleteNode(ITreeNode node);
}
```

TPEntity.java

```
/*********************************************************************
 *
 * File     :     TPEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A generic TPTeam entity use in XML serializaion and
 *                GUI displays
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

/*********************************************************************
* File            :     TPEntity.java
*
* Description     :     A generic TPTeam entity use in XML
*                       serialization and GUI displays
*
* @author Bob Brady, rpbrady@gmail.com
* @version $Revision: 367 $
* @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $ Copyright
(c) 2007 Bob Brady
*********************************************************************/
public class TPEntity extends AbstractTreeNode {

      private static final long serialVersionUID = 1L;

      /** TPTeam folder test type */
      public static final String FOLDER = "FOLDER";
      /** TPTeam JUnit test type */
      public static final String JUNIT_TEST = "JUNIT";
      /** TPTeam test execution type */
      public static final String EXEC = "EXEC";
      /** TPTeam test execution passed verdict */
      public static final String PASS = "PASS";
      /** TPTeam test execution failed verdict */
      public static final String FAIL = "FAIL";
      /** TPTeam test execution inconclusive verdict */
      public static final String INCONCLUSIVE = "INCONCLUSIVE";
      /** TPTeam test execution error verdict */
      public static final String ERROR = "ERROR";
      /** TPTeam test execution passed verdict for execution entity */
      public static final String EXEC_PASS = "EXEC_" + PASS;
      /** TPTeam test execution failed verdict for execution entity */
      public static final String EXEC_FAIL = "EXEC_" + FAIL;
      /** TPTeam test execution error verdict for execution entity */
      public static final String EXEC_ERROR = "EXEC_" + ERROR;
      /** TPTeam test execution inconclusive verdict for execution
entity */
      public static final String EXEC_INCONCLUSIVE = "EXEC_" +
INCONCLUSIVE;
      /** Description of entity from TPTeam database */
      private String mDescription;
      /** Type of TPTeam entity: test folder, JUnit, test execution */
      private String mType;
```

```
	/**
	 * Default constructor
	 */
	public TPEntity() {

	}

	/**
	 * Constructor
	 *
	 * @param id The TPTeam ID
	 * @param name The TPTeam name
	 * @param description The TPTeam description
	 * @param type The TPTeam test type
	 */
	public TPEntity(String id, String name, String description,
String type) {
		setID(id);
		setName(name);
		mDescription = description;
		mType = type;
	}

	// Property accessors

	public String getDescription() {
		return mDescription;
	}

	public void setDescription(String description) {
		mDescription = description;
		fireNodeUpdated();
	}

	public String getType() {
		return mType;
	}

	public void setType(String type) {
		mType = type;
	}
}
```

TPEvent.java

```
/*****************************************************************
 *
 * File     :     TPEvent.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A serializable object that contains information about
 *                TPTeam CRUD and execution events
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

import java.io.Serializable;
import java.util.Hashtable;
import org.osgi.service.event.Event;

/*****************************************************************
* File            :     TPEvent.java
*
* Description     :     A serializable object that contains information
*                       about TPTeam CRUD and execution events
*
* @author Bob Brady, rpbrady@gmail.com
* @version $Revision: 367 $
* @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $ Copyright
(c) 2007 Bob Brady
*****************************************************************/
public class TPEvent implements Serializable {

      // Fields
      private static final long serialVersionUID =
7318549536622346381L;
      // Helper keys used to perform look-ups in dicitonary
      public static final String PROJECT_KEY = "PROJECT";
      public static final String PROJECT_ID_KEY = "PROJ_ID";
      public static final String PROJ_PROD_XML_KEY = "PROD_PROJ_XML";
      public static final String TEST_XML_KEY = "TEST_XML";
      public static final String TEST_TREE_XML_KEY = "TEST_TREE_XML";
      public static final String TEST_EXEC_XML_KEY = "TEST_EXEC_XML";
      public static final String TEST_PROP_XML_KEY = "TEST_PROP_XML";
      public static final String CHART_DATASET_XML_KEY =
"CHART_DATASET_XML_KEY";
      public static final String VERDICT_KEY = "VERDICT";
      public static final String TEST_NAME_KEY = "TEST_NAME";
      public static final String TEST_DESC_KEY = "TEST_DESC";
      public static final String TIMESTAMP_KEY = "TIMESTAMP";
      public static final String COMMENTS_KEY = "COMMENTS";
      /** The ID from the TPTeam Database */
      public static final String ID_KEY = "ID";
      /** The ID of the parent entity from the TPTeam database */
      public static final String PARENT_ID_KEY = "PARENT_ID";
      public static final String ECFID_KEY = "ECFID";
      public static final String STATUS_KEY = "STATUS";
      /** The ECF ID of the intended recipient */
      public static final String SEND_TO = "SEND_TO";
```

```
	/** The ECF ID of the sender */
	public static final String FROM = "FROM";
	/** The table of key=value pairs */
	private Hashtable<String, String> mDictionary;
	/** The topic of the event */
	private String mTopic;

	/**
	 * Constructor
	 * @param event The corresponding OSGi Event
	 */
	public TPEvent(Event event) {
		mDictionary = new Hashtable<String, String>();
		for (String propName : event.getPropertyNames())
			mDictionary.put(propName, (String)
event.getProperty(propName));

		mTopic = event.getTopic();
	}

	/**
	 * Constructor
	 * @param topic the topic of the TPTeam event
	 * @param dictionary map of key=value pairs
	 */
	public TPEvent(String topic, Hashtable<String, String>
dictionary) {

		mDictionary = dictionary;
		mTopic = topic;
	}

	// Public accessors

	public void setTopic(String topic) {
		mTopic = topic;
	}

	public String getTopic() {
		return mTopic;
	}

	public Hashtable<String, String> getDictionary() {
		return mDictionary;
	}

	public String getProject() {
		return (String) mDictionary.get(PROJECT_KEY);
	}

	public String getTestName() {
		return (String) mDictionary.get(TEST_NAME_KEY);
	}

	public String getID() {
		return (String) mDictionary.get(ID_KEY);
	}
```

```
	public String getStatus() {
		return (String) mDictionary.get(STATUS_KEY);
	}

	public void setStatus(String status) {
		mDictionary.put(STATUS_KEY, status);
	}

}
```

TreeNodeModel.java

```
/*****************************************************************
 *
 * File     :     TreeNodeModel.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A container for ITreeNode objects that provides
 *                O(1) look-ups and listens for ITreeNode events for
 *                the automated updating of its model
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.model;

import java.util.HashMap;

/*****************************************************************
* File            :     TreeNodeModel.java
*
* Description     :     A container for ITreeNode objects that provides
*                       O(1) look-ups and listens for ITreeNode events
*                       for the automated updating of its model
*
* @author Bob Brady, rpbrady@gmail.com
* @version $Revision: 367 $
* @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $ Copyright
(c) 2007 Bob Brady
*****************************************************************/
public class TreeNodeModel extends HashMap<String, ITreeNode>
implements
            ITreeNodeChangeListener {

      private static final long serialVersionUID = 1L;

      // Public accessors

      public ITreeNode put(String key, ITreeNode node) {
            node.addChangeListener(this);
            return super.put(key, node);
      }

      public ITreeNode remove(String key, ITreeNode node) {
            node.removeChangeListener(this);
            return super.remove(key);
      }

      public void addNode(ITreeNode node) {
            addListenerTo(node);
            put(String.valueOf(node.getID()), node);
            for (ITreeNode child : node.getChildren()) {
                  addNode(child);
            }
      }

      public void deleteNode(ITreeNode node) {
            if (get(String.valueOf(node.getID())) != null) {
```

```
                removeListenerFrom(node);
                remove(String.valueOf(node.getID()));
                for (ITreeNode child : node.getChildren()) {
                      deleteNode(child);
                }
          }
    }

    public void updateNode(ITreeNode node) {
          // NOOP
    }

    /**
     * Removes this TreeNodeModel as a listener
     * from all contained nodes, then clears its
     * HashMap
     */
    public void clear() {
          for (String key : keySet()) {
                removeListenerFrom(get(key));
          }
          super.clear();
    }

    /**
     * Removes this TreeNodeModel as a listener
     * from the given ITreeNode
     *
     * @param node the ITreeNode
     */
    protected void removeListenerFrom(ITreeNode node) {
          node.removeChangeListener(this);
          for (ITreeNode child : node.getChildren()) {
                removeListenerFrom(child);
          }
    }

    /**
     * Adds this TreeNodeModel as a listener
     * to the given ITreeNode
     *
     * @param node the ITreeNode
     */
    protected void addListenerTo(ITreeNode node) {
          node.addChangeListener(this);
          for (ITreeNode child : node.getChildren()) {
                addListenerTo(child);
          }
    }

}
```

package edu.harvard.fas.rbrady.tpteam.tpbridge.xml

ChartDataSetXML.java

```
/*************************************************************************
 *
 * File     :     ChartDataSetXML.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     XML Serialization utility methods for ChartDataSet
 *                ojbects
 *
 *************************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.xml;

import java.io.ByteArrayOutputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;
import org.apache.commons.betwixt.io.BeanReader;
import org.apache.commons.betwixt.io.BeanWriter;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
/*************************************************************************
 * File           : ChartDataSetXML.java
 *
 * Description    : XML Serialization utility methods for ChartDataSet
 *                ojbects
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *************************************************************************/
public class ChartDataSetXML {

      /**
       * Serializes a single ChartDataSet object into an XML String
       *
       * @param chartDataSet
       *            the ChartDataSet object to be serialized
       * @return the XML String of the serialization
       */
      public static String getXML(ChartDataSet chartDataSet) {
            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.write("chartDataSet", chartDataSet);
                  bWriter.flush();
                  bWriter.close();
```

```
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
      }

      /**
       * Reconstitutes a ChartDataSet object from its XML serialization
String
       *
       * @param dataSetXML
       *            the String XML serializaton
       * @return the reconstituted ChartDataSet object
       */
      public static ChartDataSet getDataSetFromXML(String dataSetXML) {
            ChartDataSet dataSet = null;
            try {
                  BeanReader reader = new BeanReader();
                  reader.registerBeanClass("chartDataSet",
ChartDataSet.class);
                  StringReader xmlReader = new
StringReader(dataSetXML);
                  dataSet = (ChartDataSet) reader.parse(xmlReader);
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return dataSet;
      }

      /**
       * Serializes a List of ChartDataSet objects into an XML String
       *
       * @param dataSetList
       *            the List of ChartDataSet objects to be serialized
       * @return the XML String of the serialization
       */
      public static String getListXML(List<ChartDataSet> dataSetList) {

            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.writeXmlDeclaration("<chartDataSets>");
                  for (ChartDataSet dataSet : dataSetList)
                        bWriter.write("chartDataSet", dataSet);
                  bWriter.writeXmlDeclaration("</chartDataSets>");
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
```

```
	}

	/**
	 * Reconstitutes a List of ChartDataSet objects from its
	 * XML serialization String
	 *
	 * @param dataSetListXML the XML serialization String
	 * @return An array of reconstituted ChartDataSet objects
	 */
	@SuppressWarnings("unchecked")
	public static ChartDataSet[] getDataSetsFromXML(String
dataSetListXML) {
		ArrayList<ChartDataSet> dataSetList = null;
		try {
			BeanReader reader = new BeanReader();
			reader.registerBeanClass("chartDataSets",
ArrayList.class);
			reader.registerBeanClass("chartDataSet",
ChartDataSet.class);
			StringReader xmlReader = new
StringReader(dataSetListXML);
			dataSetList = (ArrayList<ChartDataSet>)
reader.parse(xmlReader);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return (ChartDataSet[]) dataSetList
				.toArray(new ChartDataSet[dataSetList.size()]);
	}
}
```

ProjectXML.java

```
/*********************************************************************
 *
 * File     :     ProjectXML.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Contains various utility methods for the
 *                XML serialization and deserialization of Project
 *                objects.
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.xml;

import java.io.ByteArrayOutputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;
import java.util.Set;
import org.apache.commons.betwixt.io.BeanReader;
import org.apache.commons.betwixt.io.BeanWriter;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;

/*********************************************************************
 * File           :     ProjectXML.java
 *
 * Description    :     Contains various utility methods for the XML
 *                      serialization and deserialization of Project
 *                      objects.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ProjectXML {

      /**
       * Gets String XML representation of a Set of Projects in form:
       *
       * <projects> <project id="1" name="proj1"...> <product id="1"
name="prod1"
       * .../> </project> <project id="2" name="proj2"...> <product
id="2"
       * name="prod2" .../> </project> </projects>
       *
       * @param projs
       *            Set of Projects
       * @return String representation of Projects in XML
       */
      public static String getXML(Set<Project> projs) {
            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
```

```
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.writeXmlDeclaration("<projects>");
                  for (Project proj : projs)
                        bWriter.write("project", proj);
                  bWriter.writeXmlDeclaration("</projects>");
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
      }

      /**
       * Extracts a List of Projects from XML String
       *
       * @param xml
       *            String representing projects
       * @return ArrayList of project objects
       */
      @SuppressWarnings("unchecked")
      public static List<Project> getProjsFromXML(String xml) {
            ArrayList<Project> projs = null;
            try {
                  BeanReader reader = new BeanReader();
                  reader.registerBeanClass("projects",
ArrayList.class);
                  reader.registerBeanClass("project", Project.class);
                  StringReader xmlReader = new StringReader(xml);
                  projs = (ArrayList<Project>) reader.parse(xmlReader);
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return projs;
      }

}
```

TestExecutionXML.java

```
/*****************************************************************
 *
 * File     :     TestExecutionXML.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Contains various utility methods for the
 *                XML serialization and deserialization of
 *                TestExecution objects.
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.xml;

import java.io.ByteArrayOutputStream;
import java.io.StringReader;
import org.apache.commons.betwixt.io.BeanReader;
import org.apache.commons.betwixt.io.BeanWriter;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.AbstractTreeNode;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNode;
import
edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNodeChangeListener;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*****************************************************************
 * File           :     TestExecutionXML.java
 *
 * Description    :     Contains various utility methods for the XML
 *                      serialization and deserialization of
 *                      TestExecution objects.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TestExecutionXML {

      /**
       * Extracts a TestExecution TPEntity from a TPEvent
       *
       * @param tpEvent the TPEvent
       * @return the corresponding TPEntity
       */
      public static TPEntity getTPEntityFromExecEvent(TPEvent tpEvent)
      {
            TPEntity tpEntity = null;
            String id = tpEvent.getID();
            String name = getExecEntityName(tpEvent);
            String verdict =
tpEvent.getDictionary().get(TPEvent.VERDICT_KEY).toUpperCase();
            if(verdict.equals(TPEntity.PASS))
                  tpEntity = new TPEntity(id, name, null,
TPEntity.EXEC_PASS);
            else if(verdict.equals(TPEntity.FAIL))
```

```
                  tpEntity = new TPEntity(id, name, null,
TPEntity.EXEC_FAIL);
            else if(verdict.equals(TPEntity.ERROR))
                  tpEntity = new TPEntity(id, name, null,
TPEntity.EXEC_ERROR);
            else
                  tpEntity = new TPEntity(id, name, null,
TPEntity.EXEC_INCONCLUSIVE);

            String parentID =
tpEvent.getDictionary().get(TPEvent.PARENT_ID_KEY);
            TPEntity parent = new TPEntity(parentID, null, null, null);
            parent.addChild(tpEntity);
            tpEntity.setParent(parent);
            return tpEntity;
      }

      /**
       * Gets the TestExecution TPEntity name based on the
       * execution verdict, timestamp, and execution ID
       * @param tpEvent the TPEvent
       * @return the TPEntity name
       */
      private static String getExecEntityName(TPEvent tpEvent)
      {
            StringBuilder nameBuffer = new StringBuilder();

      nameBuffer.append(tpEvent.getDictionary().get(TPEvent.VERDICT_KEY
));
            nameBuffer.append(" ");

      nameBuffer.append(tpEvent.getDictionary().get(TPEvent.TIMESTAMP_K
EY));
            nameBuffer.append(" " +
tpEvent.getDictionary().get(TPEvent.ECFID_KEY));
            return nameBuffer.toString();
      }

      /**
       * Gets the String XML serialization of the TPEntity
       * @param tpEntity the TPEntity to be serialized
       * @return the String XML serialization
       */
      public static String getTPEntityXML(TPEntity tpEntity)
      {
            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.write("tpEntity", tpEntity);
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
```

```
                e.printStackTrace();
            }
            return baos.toString();
        }

        /**
         * Reconstitutes the TPEntity object from its
         * given String XML serialization
         *
         * @param entityXML the String XML serialization
         * @return the reconstituted TPEntity object
         */
        public static TPEntity getTPEntityFromXML(String entityXML)
        {
            TPEntity tpEntity = null;
            try
            {
            BeanReader reader = new BeanReader();
            reader.registerBeanClass(AbstractTreeNode.class);
            reader.registerBeanClass(ITreeNode.class);
            reader.registerBeanClass(ITreeNodeChangeListener.class);
            reader.registerBeanClass("tpEntity", TPEntity.class);
            StringReader xmlReader = new StringReader(entityXML);
            tpEntity = (TPEntity)reader.parse(xmlReader);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            return tpEntity;
        }
}
```

TestXML.java

```
/*********************************************************************
 *
 * File     :     TestXML.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Contains various utility methods for the
 *                XML serialization and deserialization of
 *                Test objects.
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbridge.xml;

import java.io.ByteArrayOutputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import org.apache.commons.betwixt.io.BeanReader;
import org.apache.commons.betwixt.io.BeanWriter;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestExecution;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.AbstractTreeNode;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNode;
import
edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNodeChangeListener;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;

/*********************************************************************
 * File           :     TestXML.java
 *
 * Description    :     Contains various utility methods for the XML
 *                      serialization and deserialization of Test
 *                      objects.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision: 367 $
 * @date $Date: 2007-06-16 16:20:45 -0400 (Sat, 16 Jun 2007) $
Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class TestXML {

      /**
       * Extracts a skeletonized Test TPEntity from the corresponding
       * TPTeam Test object
       *
       * @param test the TPTeam Test object
       * @return the skeletonized TPEntity
       */
      public static TPEntity getTPEntityFromTest(Test test) {
            TPEntity tpEntity = null;
            String id = String.valueOf(test.getId());
            String name = test.getName();
            String desc = test.getDescription();
```

```
            String isFolder = String.valueOf(test.getIsFolder());

            if (isFolder != null && isFolder.equalsIgnoreCase("Y")) {
                  tpEntity = new TPEntity(id, name, desc,
TPEntity.FOLDER);
            } else if (test.getTestType() != null
                        &&
test.getTestType().getName().equalsIgnoreCase("JUnit")) {
                  tpEntity = new TPEntity(id, name, desc,
TPEntity.JUNIT_TEST);
            }
            return tpEntity;

      }

      /**
       * Gets the String XML serialization of the
       * given TPTeam Test object
       *
       * @param test the TPTeam Test object
       * @return the String XML serialization
       */
      public static String getXML(Test test) {
            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.write("test", test);
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
      }

      /**
       * Gets the String XML serialization of a List of TPTeam
       * Test objecs
       *
       * @param tests the List of TPTeam Test objects
       * @param projName the name of the parent TPTeam
       *    test project
       * @return String XML serialization
       */
      public static String getTPEntityXML(List<Test> tests, String
projName) {
            ArrayList<ITreeNode> tpEntities = new
ArrayList<ITreeNode>();
            TPEntity rootEntity = new TPEntity("0", projName, projName,
                        TPEntity.FOLDER);

            for (Test test : tests) {
```

```
                  tpEntities.add(getTPEntity(test, rootEntity));
            }
            rootEntity.setChildren(tpEntities);

            ByteArrayOutputStream baos = null;
            try {
                  baos = new ByteArrayOutputStream();
                  BeanWriter bWriter = new BeanWriter(baos);
                  bWriter.setWriteEmptyElements(false);
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.write("tpEntity", rootEntity);
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();
      }

      /**
       * Converts a TPTeam Test object into its corresponding
       * TPEntity and adds it as a child to the given parent
       *
       * @param test the TPTeam Test
       * @param parent the TPEntity to acting as parent
       * @return the TPEntity object of the conversion
       */
      public static TPEntity getTPEntity(Test test, TPEntity parent) {
            TPEntity tpEntity = null;
            String id = String.valueOf(test.getId());
            String name = test.getName();
            String desc = null /* test.getDescription() */;
            String isFolder = String.valueOf(test.getIsFolder());
            String testTypeName = null;
            if (test.getTestType() != null)
                  testTypeName = test.getTestType().getName();

            if (isFolder != null && isFolder.equalsIgnoreCase("Y")) {
                  tpEntity = new TPEntity(id, name, desc,
TPEntity.FOLDER);
            } else if (testTypeName != null
                        && testTypeName.equalsIgnoreCase("JUnit")) {
                  tpEntity = new TPEntity(id, name, desc,
TPEntity.JUNIT_TEST);
            }
            tpEntity.setParent(parent);
            for (Test childTest : test.getChildren()) {
                  TPEntity childEntity = getTPEntity(childTest,
tpEntity);
                  tpEntity.addChild(childEntity);
            }
            return tpEntity;
      }

      /**
```

```
 * Reconstitutes a TPEntity from its String XML
 * serialization
 *
 * @param entityXML the String XML serializaiton
 * @return the resconstituted TPEntity
 */
public static TPEntity getTPEntityFromXML(String entityXML) {
      TPEntity tpEntity = null;
      try {
            BeanReader reader = new BeanReader();
            reader.registerBeanClass(AbstractTreeNode.class);
            reader.registerBeanClass(ITreeNode.class);

reader.registerBeanClass(ITreeNodeChangeListener.class);
            reader.registerBeanClass("tpEntity", TPEntity.class);
            StringReader xmlReader = new StringReader(entityXML);
            tpEntity = (TPEntity) reader.parse(xmlReader);
      } catch (Exception e) {
            e.printStackTrace();
      }
      return tpEntity;
}

/**
 * Reconstitutes a TPTeam Test object from its
 * String XML serialization
 *
 * @param testXML the String XML serialization
 * @return the reconstituted TPTeam Test object
 */
public static Test getTestFromXML(String testXML) {
      Test test = null;
      try {
            BeanReader reader = new BeanReader();
            reader.registerBeanClass("test", Test.class);
            StringReader xmlReader = new StringReader(testXML);
            test = (Test) reader.parse(xmlReader);
      } catch (Exception e) {
            e.printStackTrace();
      }
      return test;
}

/**
 * Gets String XML serialization of Test object's
 * properites, including all test executions
 *
 * @param test the Test to be serialized
 * @return the String XML serialization
 */
public static String getTestPropXML(Test test) {
      TPEntity[] tpEntities = getTPEntityTestProps(test);
      ByteArrayOutputStream baos = null;
      try {
            baos = new ByteArrayOutputStream();
            BeanWriter bWriter = new BeanWriter(baos);
            bWriter.setWriteEmptyElements(false);
```

```
                  bWriter.enablePrettyPrint();
                  bWriter.writeXmlDeclaration("<?xml version='1.0'
?>");
                  bWriter.writeXmlDeclaration("<tpEntities>");
                  for (TPEntity tpEntity : tpEntities)
                        bWriter.write("tpEntity", tpEntity);
                  bWriter.writeXmlDeclaration("</tpEntities>");
                  bWriter.flush();
                  bWriter.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return baos.toString();

      }

      /**
       * Gets an array of TPEntities corresponding to the
       * given Test's properites
       *
       * @param test the TPTeam Test
       * @return the test properties as an array of TPEntity
       */
      private static TPEntity[] getTPEntityTestProps(Test test) {
            ArrayList<TPEntity> list = new ArrayList<TPEntity>();

            addProp(list, "ID", String.valueOf(test.getId()));
            addProp(list, "Name", test.getName());
            addProp(list, "Description", test.getDescription());
            String name = test.getCreatedBy().getLastName() + ", "
                        + test.getCreatedBy().getLastName();
            addProp(list, "Created By", name);
            addProp(list, "Created Date",
test.getCreatedDate().toString());

            if (test.getModifiedBy() != null) {
                  name = test.getModifiedBy().getLastName() + ", "
                              + test.getModifiedBy().getLastName();
                  addProp(list, "Modified By", name);
                  addProp(list, "Modified Date",
test.getModifiedDate().toString());
            }
            if (test.getIsFolder() == 'Y') {
                  addProp(list, "TestType", TPEntity.FOLDER);
            } else if
(test.getTestType().getName().toUpperCase().equals(
                        TPEntity.JUNIT_TEST)) {
                  addProp(list, "TestType", TPEntity.JUNIT_TEST);
                  if (test.getJunitTests() != null)
                        addJunitProps(list, test);
            }
            if (test.getTestExecutions() != null) {
                  addExecProps(list, test);
            }
            return (TPEntity[]) list.toArray(new
TPEntity[list.size()]);
      }
```

```
      /**
       * Adds a simple TPEntity having type and description
       * properties to a list of TPEntities
       *
       * @param list the List of TPEntities
       * @param type the type property of the TPEntity to be added
       * @param desc the description property of the TPEntity to
       *    be added
       */
      private static void addProp(ArrayList<TPEntity> list, String
type,
                  String desc) {
            TPEntity tpEntity = new TPEntity();
            tpEntity.setType(type);
            if (desc != null)
                  tpEntity.setDescription(desc);
            else
                  tpEntity.setDescription("");
            list.add(tpEntity);
      }

      /**
       * Adds the JUnit properties of a given TPTeam Test
       * object to a List of TPEntities
       *
       * @param list the List of TPEntities
       * @param test the given Test object
       */
      private static void addJunitProps(ArrayList<TPEntity> list, Test
test) {
            for (JunitTest junit : test.getJunitTests()) {
                  addProp(list, "Eclipse Home",
junit.getEclipseHome());
                  addProp(list, "Eclipse Workspace",
junit.getWorkspace());
                  addProp(list, "Eclipse Project", junit.getProject());
                  addProp(list, "TPTP Testsuite",
junit.getTestSuite());
                  addProp(list, "TPTP Report Dir",
junit.getReportDir());
                  addProp(list, "TPTP Connection",
junit.getTptpConnection());
            }
      }

      /**
       * Adds the test execution properties of a given TPTeam
       * Test object to a List of TPEntities
       *
       * @param list the List of TPEntities
       * @param test the given Test object
       */
      private static void addExecProps(ArrayList<TPEntity> list, Test
test) {
```

```
            TestExecution[] testExecs =
test.getTestExecutions().toArray(new
TestExecution[test.getTestExecutions().size()]);
            Arrays.sort(testExecs);
            for (TestExecution testExec : testExecs) {
                  String type = null;
                  StringBuilder desc = new StringBuilder();
                  if (testExec.getStatus() == 'P') {
                        type = TPEntity.EXEC_PASS;
                  } else if (testExec.getStatus() == 'F') {
                        type = TPEntity.EXEC_FAIL;
                  } else if (testExec.getStatus() == 'E') {
                        type = TPEntity.EXEC_ERROR;
                  } else {
                        type = TPEntity.EXEC_INCONCLUSIVE;
                  }
                  desc.append(testExec.getExecDate().toString());
                  desc.append(" " +
testExec.getTpteamUser().getLastName() + ", "
                              +
testExec.getTpteamUser().getLastName());
                  desc.append(" (" +
testExec.getTpteamUser().getEcfId() + ")");
                  addProp(list, type, desc.toString());
            }
      }

      /**
       * Reconstitutes an array of Test object property TPEntities
       * from an XML String serialization
       *
       * @param testPropXML the String XML serialization
       * @return the array of reconstituted TPEntity objects
       */
      @SuppressWarnings("unchecked")
      public static TPEntity[] getTPEntitiesFromXML(String testPropXML)
{
            ArrayList<TPEntity> tpEntities = null;
            try {
                  BeanReader reader = new BeanReader();
                  reader.registerBeanClass("tpEntities",
ArrayList.class);
                  reader.registerBeanClass("tpEntitity",
TPEntity.class);
                  StringReader xmlReader = new
StringReader(testPropXML);
                  tpEntities = (ArrayList<TPEntity>)
reader.parse(xmlReader);
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return (TPEntity[]) tpEntities.toArray(new
TPEntity[tpEntities.size()]);
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy

Activator.java

```
/*********************************************************************
 *
 * File     :     Activator.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Controls the lifecycle of the TPBuddy Plug-in
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy;

import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.osgi.framework.BundleContext;

import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminClient;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.tpbridge.TPBridgeClient;

/*********************************************************************
 * File           :     Activator.java
 *
 * Description    :     Controls the lifecycle of the TPBuddy Plug-in
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$
 * Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class Activator extends AbstractUIPlugin {

      // The plug-in ID
      public static final String PLUGIN_ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy";

      // The shared instance
      private static Activator plugin;

      private EventAdminClient mEventAdminClient;

      private EventAdminHandler mEventAdminHandler;

      private TPBridgeClient mTPBridgeClient;

      private String mProjID;

      private String mProjName;

      public static final String APPLICATION_WINDOW_TITLE = "TPBuddy";
```

```
	public static final int APPLICATION_WINDOW_SIZE_X = 600;
	public static final int APPLICATION_WINDOW_SIZE_Y = 400;
	public static final String CONNECT_WIZARD_PAGE_TITLE = "Connect
to XMPP Server";
	public static final String CONNECT_WIZARD_PAGE_DESCRIPTION =
"Enter user id below and login";



	/**
	 * The constructor
	 */
	public Activator() {
		plugin = this;
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.BundleC
ontext)
	 */
	public void start(BundleContext context) throws Exception {
		super.start(context);

		mEventAdminHandler = new EventAdminHandler(context);
		mEventAdminClient = new EventAdminClient(context);
		mTPBridgeClient = new TPBridgeClient(context);

	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.BundleCo
ntext)
	 */
	public void stop(BundleContext context) throws Exception {
		plugin = null;
		super.stop(context);
	}

	/**
	 * Returns the shared instance
	 *
	 * @return the shared instance
	 */
	public static Activator getDefault() {
		return plugin;
	}

	/**
	 * Returns an image descriptor for the image file at the given
plug-in
```

```
	 * relative path
	 *
	 * @param path
	 *            the path
	 * @return the image descriptor
	 */
	public static ImageDescriptor getImageDescriptor(String path) {
		return imageDescriptorFromPlugin(PLUGIN_ID, path);
	}

	// Public accessors

	public  EventAdminClient getEventAdminClient() {
		return mEventAdminClient;
	}

	public  EventAdminHandler getEventAdminHandler() {
		return mEventAdminHandler;
	}

	public TPBridgeClient getTPBridgeClient()
	{
		return mTPBridgeClient;
	}

	public void setProjID(String projID)
	{
		mProjID = projID;
	}

	public String getProjID()
	{
		return mProjID;
	}

	public void setProjName(String projName)
	{
		mProjName = projName;
	}

	public String getProjName()
	{
		return mProjName;
	}
}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions

EventHistoryAction.java

```
/*****************************************************************
 *
 * File     :     EventHistoryAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the Event History View
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.views.EventHistoryView;

/*****************************************************************
*
 * File           :     EventHistoryAction.java
 *
 * Description    :     Action that displays the Event History View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady

*****************************************************************/
public class EventHistoryAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      // Fields

      private IWorkbenchWindow window;

      private final String viewID = EventHistoryView.ID;

      public void run() {
      }


      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
```

```
	 * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
	 */
	public void dispose() {
	}

	/**
	 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
	 */
	public void init(IWorkbenchWindow window) {
		this.window = window;
	}

	/**
	 * Shows the Event History View
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		if (window != null) {
			try {
				window.getActivePage().showView(viewID);
			} catch (PartInitException e) {
				MessageDialog.openError(window.getShell(),
"Error",
					"Error opening view:" +
e.getMessage());
			}
		}
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

ExitAction.java

```
/*********************************************************************
 *
 * File     :     ExitAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that quits the TPBuddy RCP application
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;
import org.eclipse.ui.actions.ActionFactory;

/*********************************************************************
 * File           :     ExitAction.java
 *
 * Description    :     Action that quits the TPBuddy RCP application
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *********************************************************************/
public class ExitAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      private IWorkbenchWindow window;

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
       * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
       */
      public void init(IWorkbenchWindow window) {
            this.window = window;
      }
```

```
	/**
	 * Quits the TPBuddy RCP application
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		ActionFactory.QUIT.create(window).run();
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

ICommandIds.java

```
/*****************************************************************
 *
 * File     :     ICommandsIds.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Convenience interface providing constants
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

/*****************************************************************
 * File           :     ICommandsIds.java
 *
 * Description    :     Convenience interface providing constants for
 *                      key bindings
 *
 * @see org.eclipse.jface.action.IAction#setActionDefinitionId(String)
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *****************************************************************/
public interface ICommandIds {

    public static final String CMD_OPEN =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.open";
    public static final String CMD_OPEN_MESSAGE =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.openMessage";

}
```

LoginAction.java

```
/*********************************************************************
 *
 * File     :     LoginAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the XMPPConnectionWizard dialog
 *                so that user may login to TPTeam
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.ecf.core.ContainerCreateException;
import org.eclipse.ecf.core.ContainerFactory;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.wizard.WizardDialog;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.wizard.XMPPConnectWizard;

/*********************************************************************
 * File           :     LoginAction.java
 *
 * Description    :     Action that displays the XMPPConnectionWizard
 *                      dialog so that user may login to TPTeam
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *********************************************************************/
public class LoginAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
```

```
       * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
       */
      public void init(IWorkbenchWindow window) {
      }

      /**
       * Opens an XMPPConnectWizard dialog so that a user may login to
TPTeam
       *
       * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
       */
      public void run(IAction action) {
            // Create the wizard dialog
            IContainer container = null;
            try {
                  container =
ContainerFactory.getDefault().createContainer(
                              XMPPConnectWizard.CONTAINER_TYPE);
            } catch (ContainerCreateException e) {

      MessageDialog.openError(getWorkbench().getActiveWorkbenchWindow()
                              .getShell(), "Create Error",
                              "Could not create XMPP
container.\n\nError: "
                                          + e.getLocalizedMessage());
            }
            XMPPConnectWizard connectWizard = new XMPPConnectWizard();
            connectWizard.init(getWorkbench(), container);
            WizardDialog dialog = new WizardDialog(getWorkbench()
                        .getActiveWorkbenchWindow().getShell(),
connectWizard);
            // Open the wizard dialog
            dialog.open();

      }

      /*
       * (non-Javadoc)
       *
       * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
       *      org.eclipse.jface.viewers.ISelection)
       */
      public void selectionChanged(IAction action, ISelection
selection) {
      }
}
```

ProjectReportAction.java

```
/*********************************************************************
 *
 * File     :     ProjectReportAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the Project Report View
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.views.ReportView;

/*********************************************************************
 * File           :     ProjectReportAction.java
 *
 * Description    :     Action that displays the Project Report View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *********************************************************************/
public class ProjectReportAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      private IWorkbenchWindow window;

      /** The ID of the ReportView */
      private final String viewID = ReportView.ID;

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
```

```
	 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
	 */
	public void init(IWorkbenchWindow window) {
		this.window = window;
	}

	/**
	 * Opens the ReportView
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		if (window != null) {
			try {
				window.getActivePage().showView(viewID);
			} catch (PartInitException e) {
				MessageDialog.openError(window.getShell(),
"Error",
					"Error opening view:" +
e.getMessage());
			}
		}
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

ProjectViewAction.java

```
/*************************************************************************
 *
 * File     :     ProjectViewAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the Projects View
 *
 *************************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;

import edu.harvard.fas.rbrady.tpteam.tpbuddy.views.ProjectView;

/*************************************************************************
 * File           :     ProjectViewAction.java
 *
 * Description    :     Action that displays the Projects View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *************************************************************************/
public class ProjectViewAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      private IWorkbenchWindow window;

      /** The ID of the ProjectView */
      private final String viewID = ProjectView.ID;

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
```

```
	 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
	 */
	public void init(IWorkbenchWindow window) {
		this.window = window;
	}

	/**
	 * Opens the Projects View
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		if (window != null) {
			try {
				window.getActivePage().showView(viewID);
			} catch (PartInitException e) {
				MessageDialog.openError(window.getShell(),
"Error",
					"Error opening view:" +
e.getMessage());
			}
		}
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

TestDetailAction.java

```
/*********************************************************************
 *
 * File     :     TestDetailAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the Test Details View
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;

import edu.harvard.fas.rbrady.tpteam.tpbuddy.views.DetailView;

/*********************************************************************
 * File           :     TestDetailAction.java
 *
 * Description    :     Action that displays the Test Details View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady

*********************************************************************/
public class TestDetailAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      private IWorkbenchWindow window;

      /** The ID of the DetailView */
      private final String viewID = DetailView.ID;

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
```

```
	 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
	 */
	public void init(IWorkbenchWindow window) {
		this.window = window;
	}

	/**
	 * Opens the Test Details view
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		if (window != null) {
			try {
				window.getActivePage().showView(viewID);
			} catch (PartInitException e) {
				MessageDialog.openError(window.getShell(),
"Error",
					"Error opening view:" +
e.getMessage());
			}
		}
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

TestTreeAction.java

```
/*********************************************************************
 *
 * File     :     TestTreeAction.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Action that displays the Test Tree View
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.actions;

import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.ui.IWorkbench;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionDelegate;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.views.TestView;

/*********************************************************************
 * File           :     TestTreeAction.java
 *
 * Description    :     Action that displays the Test Tree View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *
*********************************************************************/
public class TestTreeAction extends ActionDelegate implements
            IWorkbenchWindowActionDelegate {

      private IWorkbenchWindow window;
      /** The ID of the TestView */
      private final String viewID = TestView.ID;

      public void run() {
      }

      protected IWorkbench getWorkbench() {
            return PlatformUI.getWorkbench();
      }

      /**
       * @see org.eclipse.ui.IWorkbenchWindowActionDelegate#dispose()
       */
      public void dispose() {
      }

      /**
```

```
	 * @see
org.eclipse.ui.IWorkbenchWindowActionDelegate#init(org.eclipse.ui.IWork
benchWindow)
	 */
	public void init(IWorkbenchWindow window) {
		this.window = window;
		}

	/**
	 * Opens the Test Tree View
	 *
	 * @see
org.eclipse.ui.IActionDelegate#run(org.eclipse.jface.action.IAction)
	 */
	public void run(IAction action) {
		if(window != null) {
			try {
				window.getActivePage().showView(viewID);
			} catch (PartInitException e) {
				MessageDialog.openError(window.getShell(),
"Error", "Error opening view:" + e.getMessage());
			}
		}
	}

	/**
	 * @see
org.eclipse.ui.IActionDelegate#selectionChanged(org.eclipse.jface.actio
n.IAction,
	 *	org.eclipse.jface.viewers.ISelection)
	 */
	public void selectionChanged(IAction action, ISelection
selection) {
	}
}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts

AbstractChart.java

```
/*********************************************************************
 *
 * File     :     AbstractChart.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Base class for all TPTeam Chart objects
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts;

import java.awt.Color;
import java.util.HashMap;
import org.jfree.chart.JFreeChart;
import org.jfree.data.general.AbstractDataset;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;

/*********************************************************************
 * File           :     AbstractChart.java
 *
 * Description    :     Base class for all TPTeam Chart objects
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 * *******************************************************************/
public abstract class AbstractChart {

      /** Static chart used for fast rendering */
      protected static AbstractChart mChart = null;
      /** Map of all child objects */
      protected static HashMap<Class, AbstractChart> mCharts = new
HashMap<Class, AbstractChart>();
      /** Pass test verdict */
      public static final String PASS = "Pass";
      /** Fail test verdict */
      public static final String FAIL = "Fail";
      /** Error test verdict */
      public static final String ERR = "Error";
      /** Inconclusive test verdict */
      public static final String INCONCL = "Inconclusive";
      /** Test never run */
      public static final String NOTEXEC = "Not Executed";
      /** Color representing failed test executions */
      public static final Color RED = new Color(255, 0, 0);
      /** Color representing passed test executions */
      public static final Color GREEN = new Color(34, 139, 34);
      /** Color representing inconclusive test executions */
      public static final Color BLUE = new Color(0, 0, 255);
      /** Color representing test executions stopped due to errors */
      public static final Color YELLOW = new Color(255, 255, 0);
```

```
	/** Color representin tests never executed */
	public static final Color PINK = new Color(255, 20, 147);

	public abstract AbstractDataset createDataset();

	public abstract AbstractDataset createDataset(ChartDataSet[]
dataSet);

	public abstract JFreeChart createChart();

	public abstract JFreeChart createChart(ChartDataSet[] dataSets,
String projName);

	/**
	 * Returns the static instance of this chart
	 * @return the chart instance
	 */
	public static AbstractChart getInstance()
	{
		return null;
	}

}
```

BarChart.java

```
/*********************************************************************
 *
 * File     :     BarChart.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     TPTeam implementation of a bar chart showing the
 *                project distribution of test executions by TPTeam
 *                user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts;

import java.awt.Color;
import java.util.Arrays;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.BarRenderer;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.AbstractDataset;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataPoint;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;

/*********************************************************************
 * File           :     BarChart.java
 *
 * Description    :     TPTeam implementation of a bar chart showing
 *                      the distribution of project test executions by
 *                      TPTeam user
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *********************************************************************/
public class BarChart extends AbstractChart {

      /**
       * Default Constructor
       */
      private BarChart() {
      }

      /**
       * Gets the static instance of the BarChart for
       * faster rendering time
       *
       * @return the BarChart instance
       */
```

```
	public static synchronized AbstractChart getInstance() {
		if(mCharts.get(BarChart.class) == null)
		{
			mCharts.put(BarChart.class, new BarChart());
		}
		return mCharts.get(BarChart.class);
	}

	/**
	 * Returns a sample dataset.
	 *
	 * @return The dataset.
	 */
	public AbstractDataset createDataset() {

		// row keys...
		String series1 = PASS;
		String series2 = FAIL;
		String series3 = ERR;
		String series4 = INCONCL;

		// column keys...
		String user1 = "User 1";
		String user2 = "User 2";
		String user3 = "User 3";
		String user4 = "User 4";

		// create the dataset...
		DefaultCategoryDataset dataset = new
DefaultCategoryDataset();

		dataset.addValue(1.0, series1, user1);
		dataset.addValue(4.0, series1, user2);
		dataset.addValue(3.0, series1, user3);
		dataset.addValue(5.0, series1, user4);

		dataset.addValue(5.0, series2, user1);
		dataset.addValue(7.0, series2, user2);
		dataset.addValue(6.0, series2, user3);
		dataset.addValue(8.0, series2, user4);

		dataset.addValue(4.0, series3, user1);
		dataset.addValue(3.0, series3, user2);
		dataset.addValue(2.0, series3, user3);
		dataset.addValue(3.0, series3, user4);

		dataset.addValue(1.0, series4, user1);
		dataset.addValue(3.0, series4, user2);
		dataset.addValue(5.0, series4, user3);
		dataset.addValue(3.0, series4, user4);

		return dataset;

	}

	public AbstractDataset createDataset(ChartDataSet[] dataSets) {
```

```
            DefaultCategoryDataset dataset = new
DefaultCategoryDataset();
            Arrays.sort(dataSets);
            for (ChartDataSet tpTeamDataSet : dataSets) {
                  TpteamUser user = tpTeamDataSet.getUser();
                  String userName = user.getLastName() + ", " +
user.getFirstName();
                  ChartDataPoint tpTeamDataPoints = tpTeamDataSet
                              .getChartDataPoints().get(0);
                  dataset.addValue(tpTeamDataPoints.getPass(), PASS,
userName);
                  dataset.addValue(tpTeamDataPoints.getFail(), FAIL,
userName);
                  dataset.addValue(tpTeamDataPoints.getError(), ERR,
userName);
                  dataset.addValue(tpTeamDataPoints.getInconcl(),
INCONCL, userName);
                  dataset.addValue(tpTeamDataPoints.getNotExec(),
NOTEXEC, userName);
            }
            return dataset;
      }

      /**
       * Creates a sample chart.
       *
       * @param dataset
       *            the dataset.
       *
       * @return The chart.
       */
      public JFreeChart createChart() {

            CategoryDataset dataset = (CategoryDataset)
createDataset();

            // create the chart...
            JFreeChart chart = ChartFactory.createBarChart(
                        "Project Users Overview", // title
                        "User", // domain axis label
                        "Number of Tests", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips
                        false // URLs
                        );

            // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...

            // set the background color for the chart...
            chart.setBackgroundPaint(Color.white);

            // get a reference to the plot for further customisation...
            CategoryPlot plot = chart.getCategoryPlot();
            plot.setBackgroundPaint(Color.lightGray);
            plot.setDomainGridlinePaint(Color.white);
```

```
            plot.setDomainGridlinesVisible(true);
            plot.setRangeGridlinePaint(Color.white);

            // set the range axis to display integers only...
            final NumberAxis rangeAxis = (NumberAxis)
plot.getRangeAxis();

      rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits(
));

            // disable bar outlines...
            BarRenderer renderer = (BarRenderer) plot.getRenderer();
            renderer.setSeriesPaint(0, GREEN);
            renderer.setSeriesPaint(1, RED);
            renderer.setSeriesPaint(2, YELLOW);
            renderer.setSeriesPaint(3, BLUE);
            renderer.setDrawBarOutline(false);

            CategoryAxis domainAxis = plot.getDomainAxis();
            domainAxis.setCategoryLabelPositions(CategoryLabelPositions
                        .createUpRotationLabelPositions(Math.PI /
6.0));
            // OPTIONAL CUSTOMISATION COMPLETED.

            return chart;

      }

      /**
       * Creates a bar chart of test execution status for
       * each user in a project
       *
       * @param dataSets the chart data points
       * @param projName the name of the test project
       * @return the bar chart
       */
      public JFreeChart createChart(ChartDataSet[] dataSets, String
projName) {

            CategoryDataset dataset = (CategoryDataset)
createDataset(dataSets);

            // create the chart...
            JFreeChart chart = ChartFactory.createBarChart("Project " +
projName
                        + " Users Overview", // title
                        "User", // domain axis label
                        "Number of Tests", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips
                        false // URLs
                        );

            // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
```

```
            // set the background color for the chart...
            chart.setBackgroundPaint(Color.white);

            // get a reference to the plot for further customisation...
            CategoryPlot plot = chart.getCategoryPlot();
            plot.setBackgroundPaint(Color.lightGray);
            plot.setDomainGridlinePaint(Color.white);
            plot.setDomainGridlinesVisible(true);
            plot.setRangeGridlinePaint(Color.white);

            // set the range axis to display integers only...
            final NumberAxis rangeAxis = (NumberAxis)
plot.getRangeAxis();

      rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits(
));

            // disable bar outlines...
            BarRenderer renderer = (BarRenderer) plot.getRenderer();
            renderer.setSeriesPaint(0, GREEN);
            renderer.setSeriesPaint(1, RED);
            renderer.setSeriesPaint(2, YELLOW);
            renderer.setSeriesPaint(3, BLUE);
            renderer.setSeriesPaint(4, PINK);
            renderer.setDrawBarOutline(false);

            CategoryAxis domainAxis = plot.getDomainAxis();
            domainAxis.setCategoryLabelPositions(CategoryLabelPositions
                        .createUpRotationLabelPositions(Math.PI /
6.0));
            // OPTIONAL CUSTOMISATION COMPLETED.

            return chart;

      }
}
```

LineChart.java

```
/*********************************************************************
 *
 * File     :     LineChart.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     TPTeam implementation of a lne chart showing the
 *                time history of test executions for the given
 *                project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts;

import java.awt.Color;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.StandardXYItemRenderer;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.data.general.AbstractDataset;
import org.jfree.data.time.Day;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataPoint;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;

/*********************************************************************
 * File           :     LineChart.java
 *
 * Description    :     TPTeam implementation of a lne chart showing
 *                      the time history of test executions for the
 *                      given project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *********************************************************************/
public class LineChart extends AbstractChart {

      /** Number of days in the time history window to be plotted */
      public static final int NUM_DAYS_IN_SERIES = 30;

      /**
       * Default Constructor
       */
      private LineChart() {
      }

      /**
```

```
       * Gets the static instance of the LineChart for
       * faster rendering time
       *
       * @return the LineChart instance
       */
      public static synchronized AbstractChart getInstance() {
            if(mCharts.get(LineChart.class) == null)
            {
                  mCharts.put(LineChart.class, new LineChart());
            }
            return mCharts.get(LineChart.class);
      }

      /**
       * Creates a chart.
       *
       * @param dataset
       *            a dataset.
       *
       * @return A chart.
       */
      public JFreeChart createChart() {
            XYDataset dataset = (XYDataset) createDataset();
            JFreeChart chart = ChartFactory.createTimeSeriesChart(
                        "Project Historical Status", "Date", "Number of
Tests",
                        dataset, true, true, false);
            chart.setBackgroundPaint(Color.white);
            XYPlot plot = chart.getXYPlot();
            plot.setBackgroundPaint(Color.lightGray);
            plot.setDomainGridlinePaint(Color.white);
            plot.setRangeGridlinePaint(Color.white);

            plot.setDomainCrosshairVisible(true);
            plot.setRangeCrosshairVisible(true);
            XYItemRenderer renderer = plot.getRenderer();
            renderer.setSeriesPaint(0, GREEN);
            renderer.setSeriesPaint(1, RED);
            renderer.setSeriesPaint(2, YELLOW);
            renderer.setSeriesPaint(3, BLUE);
            if (renderer instanceof StandardXYItemRenderer) {
                  StandardXYItemRenderer rr = (StandardXYItemRenderer)
renderer;
                  // rr.setPlotShapes(true);
                  rr.setShapesFilled(true);
                  rr.setItemLabelsVisible(true);
            }
            DateAxis axis = (DateAxis) plot.getDomainAxis();
            axis.setDateFormatOverride(new SimpleDateFormat("MMM-dd-
yy"));

            return chart;
      }

      public JFreeChart createChart(ChartDataSet[] dataSet, String
projName) {
            XYDataset dataset = (XYDataset) createDataset(dataSet);
```

```
            JFreeChart chart =
ChartFactory.createTimeSeriesChart("Project "
                        + projName + " Historical Status", "Date",
"Number of Tests",
                        dataset, true, true, false);
            chart.setBackgroundPaint(Color.white);
            XYPlot plot = chart.getXYPlot();
            plot.setBackgroundPaint(Color.lightGray);
            plot.setDomainGridlinePaint(Color.white);
            plot.setRangeGridlinePaint(Color.white);

            plot.setDomainCrosshairVisible(true);
            plot.setRangeCrosshairVisible(true);
            XYItemRenderer renderer = plot.getRenderer();
            renderer.setSeriesPaint(0, GREEN);
            renderer.setSeriesPaint(1, RED);
            renderer.setSeriesPaint(2, YELLOW);
            renderer.setSeriesPaint(3, BLUE);
            renderer.setSeriesPaint(4, PINK);
            if (renderer instanceof StandardXYItemRenderer) {
                  StandardXYItemRenderer rr = (StandardXYItemRenderer)
renderer;
                  // rr.setPlotShapes(true);
                  rr.setShapesFilled(true);
                  rr.setItemLabelsVisible(true);
            }
            DateAxis axis = (DateAxis) plot.getDomainAxis();
            axis.setDateFormatOverride(new SimpleDateFormat("MMM-dd-
yy"));

            return chart;
      }

      /**
       * Helper method to create a sample time series
       * @param seriesName
       * @return
       */
      private TimeSeries createTimeSeries(String seriesName) {
            Calendar cal = Calendar.getInstance();
            Date today = new Date();
            TimeSeries timeSeries = new TimeSeries(seriesName,
Day.class);
            for (int idx = 0; idx < NUM_DAYS_IN_SERIES; idx++) {
                  cal.setTime(today);
                  cal.add(Calendar.DATE, -idx);
                  if (seriesName.equalsIgnoreCase(PASS))
                        timeSeries.add(new Day(cal.getTime()), 130 -
idx);
                  else if (seriesName.equalsIgnoreCase(FAIL))
                        timeSeries.add(new Day(cal.getTime()), 100 +
idx);
                  else if (seriesName.equalsIgnoreCase(ERR))
                        timeSeries.add(new Day(cal.getTime()), 10);
                  else if (seriesName.equalsIgnoreCase(INCONCL))
                        timeSeries.add(new Day(cal.getTime()), 5);
```

```
            }
            return timeSeries;

      }

      /**
       * Helper method to create the chart time series
       *
       * @param dataSet the chart data set
       * @param seriesName the name of the time series
       * @return a JFreeChart representation of the time series
       */
      private TimeSeries createTimeSeries(ChartDataSet dataSet, String
seriesName) {
            TimeSeries timeSeries = new TimeSeries(seriesName,
Day.class);
            List<ChartDataPoint> dataPoints =
dataSet.getChartDataPoints();
            for (ChartDataPoint dataPoint : dataPoints) {
                  if (seriesName.equalsIgnoreCase(PASS))
                        timeSeries.add(new Day(dataPoint.getDate()),
dataPoint
                                    .getPass());
                  else if (seriesName.equalsIgnoreCase(FAIL))
                        timeSeries.add(new Day(dataPoint.getDate()),
dataPoint
                                    .getFail());
                  else if (seriesName.equalsIgnoreCase(ERR))
                        timeSeries.add(new Day(dataPoint.getDate()),
dataPoint
                                    .getError());
                  else if (seriesName.equalsIgnoreCase(INCONCL))
                        timeSeries.add(new Day(dataPoint.getDate()),
dataPoint
                                    .getInconcl());
                  else if (seriesName.equalsIgnoreCase(NOTEXEC))
                        timeSeries.add(new Day(dataPoint.getDate()),
dataPoint
                                    .getNotExec());
            }
            return timeSeries;
      }

      /**
       * Creates a dataset, consisting of two series of monthly data.
       *
       * @return the dataset.
       */
      public AbstractDataset createDataset() {
            TimeSeries passSeries = createTimeSeries(PASS);
            TimeSeries failSeries = createTimeSeries(FAIL);
            TimeSeries errSeries = createTimeSeries(ERR);
            TimeSeries incSeries = createTimeSeries(INCONCL);
            TimeSeriesCollection dataset = new TimeSeriesCollection();
            dataset.addSeries(passSeries);
            dataset.addSeries(failSeries);
            dataset.addSeries(errSeries);
```

```
            dataset.addSeries(incSeries);

            return dataset;
      }

      public AbstractDataset createDataset(ChartDataSet[] dataSet) {
            TimeSeries passSeries = createTimeSeries(dataSet[0], PASS);
            TimeSeries failSeries = createTimeSeries(dataSet[0], FAIL);
            TimeSeries errSeries = createTimeSeries(dataSet[0], ERR);
            TimeSeries incSeries = createTimeSeries(dataSet[0],
INCONCL);
            TimeSeries notExecSeries = createTimeSeries(dataSet[0],
NOTEXEC);
            TimeSeriesCollection dataset = new TimeSeriesCollection();
            dataset.addSeries(passSeries);
            dataset.addSeries(failSeries);
            dataset.addSeries(errSeries);
            dataset.addSeries(incSeries);
            dataset.addSeries(notExecSeries);
            return dataset;
      }

}
```

PieChart.java

```
/*************************************************************************
 *
 * File     :     PieChart.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     TPTeam implementation of a pie chart showing the
 *                top-level current snapshot of test execution
 *                status for the given project
 *
 *************************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.charts;

import java.util.List;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot;
import org.jfree.data.general.AbstractDataset;
import org.jfree.data.general.DefaultPieDataset;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataPoint;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;

/*************************************************************************
 * File           :     PieChart.java
 *
 * Description    :     TPTeam implementation of a pie chart showing
 *                      the top-level current snapshot of test
 *                      execution status for the given project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *************************************************************************/
public class PieChart extends AbstractChart {

      /**
       * Default Constructor
       */
      private PieChart() {
      }

      /**
       * Gets the static instance of the PieChart for
       * faster rendering time
       *
       * @return the PieChart instance
       */
      public static synchronized AbstractChart getInstance() {
            if(mCharts.get(PieChart.class) == null)
            {
                  mCharts.put(PieChart.class, new PieChart());
            }
            return mCharts.get(PieChart.class);
      }
```

```
	/**
	 * Returns a sample dataset.
	 *
	 * @return The dataset.
	 */
	public AbstractDataset createDataset() {
		DefaultPieDataset dataset = new DefaultPieDataset();
		dataset.setValue(PASS, 75.0);
		dataset.setValue(FAIL, 20);
		dataset.setValue(ERR, 4);
		dataset.setValue(INCONCL, 0);
		dataset.setValue(NOTEXEC, 3);
		return dataset;
	}

	/**
	 * Gets the data set in JFreeChart format
	 *
	 * @param dataSet the TPTeam set of data points
	 * @return the data points in JFreeChart abstract form
	 */
	public AbstractDataset createDataset(ChartDataSet[] dataSet) {
		List<ChartDataPoint> dataPoints =
dataSet[0].getChartDataPoints();
		ChartDataPoint dataPoint = dataPoints.get(0);

		DefaultPieDataset dataset = new DefaultPieDataset();
		dataset.setValue(PASS, dataPoint.getPass());
		dataset.setValue(FAIL, dataPoint.getFail());
		dataset.setValue(ERR, dataPoint.getError());
		dataset.setValue(INCONCL, dataPoint.getInconcl());
		dataset.setValue(NOTEXEC, dataPoint.getNotExec());
		return dataset;
	}

	/**
	 * Creates a sample chart.
	 *
	 * @param dataset
	 *            the dataset.
	 *
	 * @return The chart.
	 */
	public JFreeChart createChart() {

		DefaultPieDataset dataset = (DefaultPieDataset)
createDataset();

		// create the chart...
		JFreeChart chart = ChartFactory.createPieChart(
				"Project Test Execution Overview\nTotal Tests =
5", dataset,
				true, // legend?
				true, // tooltips?
				false // URLs?
				);
```

```
            // Set custom settings
            PiePlot plot = (PiePlot) chart.getPlot();
            plot.setSectionPaint(PASS, GREEN);
            plot.setSectionPaint(FAIL, RED);
            plot.setSectionPaint(ERR, YELLOW);
            plot.setSectionPaint(INCONCL, BLUE);
            plot.setSectionPaint(NOTEXEC, PINK);

            return chart;
      }

      /**
       * Creates a pie chart sliced by test execution status
       *
       * @param dataSet the set of data points
       * @param projName the name of the TPTeam test project
       * @return a JFreeChart pie chart
       */
      public JFreeChart createChart(ChartDataSet[] dataSet, String
projName) {

            DefaultPieDataset dataset = (DefaultPieDataset)
createDataset(dataSet);
            int totalTests = 0;
            for (int idx = 0; idx < dataset.getItemCount(); idx++)
                  totalTests += dataset.getValue(idx).intValue();

            // create the chart...
            JFreeChart chart = ChartFactory.createPieChart(projName
                        + " Test Plan Overview\nTotal Tests: " +
totalTests, dataset,
                        true, // legend?
                        true, // tooltips?
                        false // URLs?
                        );

            // Set custom settings
            PiePlot plot = (PiePlot) chart.getPlot();
            plot.setSectionPaint(PASS, GREEN);
            plot.setSectionPaint(FAIL, RED);
            plot.setSectionPaint(ERR, YELLOW);
            plot.setSectionPaint(INCONCL, BLUE);
            plot.setSectionPaint(NOTEXEC, PINK);
            return chart;
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs

AddFolderDialog.java

```
/*****************************************************************
 *
 * File     :     AddFolderDialog.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a dialog for users to enter data for new
 *                test folder creation
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs;

import org.eclipse.jface.dialogs.TitleAreaDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestType;

/*****************************************************************
 * File           :     AddFolderDialog.java
 *
 * Description    :     Provides a dialog for users to enter data for
 *                      new test folder creation
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *****************************************************************/
public class AddFolderDialog extends TitleAreaDialog {
      /** ID for OK, large integer used to avoid system conflicts */
      public static final int OK = 9999;
      /** dialog OK button */
      private Button mOKBtn;
      /** Test folder name */
      private Text mName;
      /** Test folder description */
      private Text mDescription;
      /** Stubbed Test object to represent test folder */
      private Test mTestStub;
      /** TPTeam database ID of parent test folder */
      private int mParentID;
```

```
/**
 * Constructor
 *
 * @param shell -
 *            Containing shell
 */
public AddFolderDialog(Shell shell, int parentID) {
      super(shell);
      mParentID = parentID;
}

/**
 * Complete the dialog with
 *      a title and a message
 * @see org.eclipse.jface.window.Window#create()
 */
public void create() {
      super.create();
      setTitle("Add Test Folder");
      getShell().setSize(600, 250);
}

/**
 * Fill center area of the dialog
 * @see org.eclipse.jface.dialogs.Dialog#
 *      createDialogArea(org.eclipse.swt.widgets.Composite)
 */
protected Control createDialogArea(Composite parent) {

      // Create new composite as container
      final Composite area = new Composite(parent, SWT.NULL);

      GridLayout gridLayout = new GridLayout();
      gridLayout.numColumns = 2;
      area.setLayout(gridLayout);

      GridData data = new GridData(GridData.FILL_HORIZONTAL);
      area.setLayoutData(data);

      Label nameLabel = new Label(area, SWT.NONE);
      nameLabel.setText("Name:");
      mName = new Text(area, SWT.BORDER);

      GridData nameLData = new GridData();
      nameLData.widthHint = 60;
      nameLabel.setLayoutData(nameLData);

      GridData nameData = new GridData(GridData.FILL_HORIZONTAL);
      mName.setLayoutData(nameData);

      Label descLabel = new Label(area, SWT.NONE);
      descLabel.setText("Description:");

      GridData descLData = new GridData();
      descLData.widthHint = 60;
      descLData.verticalSpan = 15;
      descLabel.setLayoutData(descLData);
```

```
            mDescription = new Text(area, SWT.MULTI | SWT.WRAP |
SWT.BORDER
                        | SWT.V_SCROLL);
            GridData descData = new GridData(GridData.FILL_BOTH);
            descData.verticalSpan = 15;
            mDescription.setLayoutData(descData);
            return area;
      }

      private boolean validate() {
            boolean returnVal = false;
            if (mName.getText() == null || mName.getText().equals(""))
{
                  setErrorMessage("Name value must not be empty.");
            } else {
                  setErrorMessage(null);
                  returnVal = true;
            }
            return returnVal;
      }

      /**
       * Replace the OK button by createButton() from Dialog
       * @see org.eclipse.jface.dialogs.Dialog#
       *
createButtonsForButtonBar(org.eclipse.swt.widgets.Composite)
       */
      protected void createButtonsForButtonBar(Composite parent) {
            // Create Ok button
            mOKBtn = createButton(parent, OK, "Ok", true);
            // Initially deactivate it
            mOKBtn.setEnabled(true);
            // Add a SelectionListener
            mOKBtn.addSelectionListener(new SelectionAdapter() {
                  public void widgetSelected(SelectionEvent e) {
                        if (validate()) {
                              mTestStub = new Test();
                              mTestStub.setName(mName.getText());

      mTestStub.setDescription(mDescription.getText());
                              mTestStub.setIsFolder('Y');
                              TestType testType = new TestType();
                              testType.setName("Folder");
                              mTestStub.setTestType(testType);
                              Test parent = new Test();
                              parent.setId(mParentID);
                              parent.addChild(mTestStub);
                              mTestStub.setParent(parent);
                              setReturnCode(OK);
                              close();
                        }
                  }
            });

            // Create Cancel button
```

```
            Button cancelButton = createButton(parent, CANCEL,
"Cancel", false);
            // Add a SelectionListener
            cancelButton.addSelectionListener(new SelectionAdapter() {
                  public void widgetSelected(SelectionEvent e) {
                        setReturnCode(CANCEL);
                        close();
                  }
            });
      }

      /**
       * Getter
       * @return the test stub representing the folder
       */
      public Test getTestStub() {
            return mTestStub;
      }

}
```

AddTestDialog.java

```
/*****************************************************************
 *
 * File     :     AddTestDialog.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a dialog for users to enter data for new
 *                test definiton creation
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs;

import org.eclipse.jface.dialogs.TitleAreaDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;

import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestType;

/*****************************************************************
 * File           :     AddTestDialog.java
 *
 * Description    :     Provides a dialog for users to enter data for
 *                      new test definition creation
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *****************************************************************/
public class AddTestDialog extends TitleAreaDialog {
      /** ID for OK, large integer used to avoid system conflicts */
      public static final int OK = 9999;
      /** dialog OK button */
      private Button mOKBtn;
      /** Test GUI type */
      private boolean mIsFolder;
      /** Name of the test definition */
      private Text mName;
      /** Description of the test definition */
      private Text mDescription;
      /** Home directory of the test */
      private Text mHome;
      /** Workspace directory location */
      private Text mWorkspace;
      /** Test project directory */
```

```
private Text mProject;
/** Test suite directory */
private Text mTestSuite;
/** Test report directory */
private Text mReportDir;
/** TPTP connection URL to be used for execution */
private Text mConnURL;
/** Stubbed version of Test object */
private Test mTestStub;
/** TPTeam database ID of test parent */
private int mParentID;

/**
 * Constructor
 *
 * @param shell -
 *            Containing shell
 */
public AddTestDialog(Shell shell, int parentID) {
      super(shell);
      mParentID = parentID;
}

/**
 * Complete the dialog with
 *      a title and a message
 * @see org.eclipse.jface.window.Window#create()
 */
public void create() {
      super.create();
      setTitle("Add Test Node");
      getShell().setSize(600, 400);
}

/**
 * Fill center area of the dialog
 * @see org.eclipse.jface.dialogs.Dialog#
 *      createDialogArea(org.eclipse.swt.widgets.Composite)
 */
protected Control createDialogArea(Composite parent) {

      // Create new composite as container
      final Composite area = new Composite(parent, SWT.NULL);

      GridLayout gridLayout = new GridLayout();
      gridLayout.numColumns = 2;
      area.setLayout(gridLayout);

      GridData data = new GridData(GridData.FILL_HORIZONTAL);
      area.setLayoutData(data);

      Label nameLabel = new Label(area, SWT.NONE);
      nameLabel.setText("Name:");
      mName = new Text(area, SWT.BORDER);

      GridData nameLData = new GridData();
      nameLData.widthHint = 60;
```

```
            nameLabel.setLayoutData(nameLData);

            GridData nameData = new GridData(GridData.FILL_HORIZONTAL);
            mName.setLayoutData(nameData);

            Label descLabel = new Label(area, SWT.NONE);
            descLabel.setText("Description:");

            GridData descLData = new GridData();
            descLData.widthHint = 60;
            descLData.verticalSpan = 15;
            descLabel.setLayoutData(descLData);

            mDescription = new Text(area, SWT.MULTI | SWT.WRAP |
SWT.BORDER
                        | SWT.V_SCROLL);
            GridData descData = new GridData(GridData.FILL_BOTH);
            descData.verticalSpan = 15;
            mDescription.setLayoutData(descData);

            if (!mIsFolder) {
                  createTestArea(area);
            }

            return area;
      }

      /**
       * Helper function to create additional test
       * definition data inputs to dialog area
       *
       * @param area the dialog area
       */
      protected void createTestArea(Composite area) {
            Label homeLabel = new Label(area, SWT.NONE);
            homeLabel.setText("Eclipse Home:");
            mHome = new Text(area, SWT.BORDER);

            Label workLabel = new Label(area, SWT.NONE);
            workLabel.setText("Eclipse Workspace:");
            mWorkspace = new Text(area, SWT.BORDER);

            Label projLabel = new Label(area, SWT.NONE);
            projLabel.setText("Eclipse Project:");
            mProject = new Text(area, SWT.BORDER);

            Label testsuiteLabel = new Label(area, SWT.NONE);
            testsuiteLabel.setText("TPTP Testsuite:");
            mTestSuite = new Text(area, SWT.BORDER);

            Label reportLabel = new Label(area, SWT.NONE);
            reportLabel.setText("TPTP Report Dir:");
            mReportDir = new Text(area, SWT.BORDER);

            Label connLabel = new Label(area, SWT.NONE);
            connLabel.setText("TPTP Conn URL:");
            mConnURL = new Text(area, SWT.BORDER);
```

```
            GridData data = new GridData();
            data.widthHint = 100;
            homeLabel.setLayoutData(data);

            data = new GridData();
            data.widthHint = 100;
            projLabel.setLayoutData(data);

            data = new GridData();
            data.widthHint = 100;
            testsuiteLabel.setLayoutData(data);

            data = new GridData();
            data.widthHint = 100;
            reportLabel.setLayoutData(data);

            data = new GridData();
            data.widthHint = 100;
            connLabel.setLayoutData(data);

            GridData data2 = new GridData(GridData.FILL_HORIZONTAL);
            mHome.setLayoutData(data2);

            data2 = new GridData(GridData.FILL_HORIZONTAL);
            mWorkspace.setLayoutData(data2);

            data2 = new GridData(GridData.FILL_HORIZONTAL);
            mProject.setLayoutData(data2);

            data2 = new GridData(GridData.FILL_HORIZONTAL);
            mTestSuite.setLayoutData(data2);

            data2 = new GridData(GridData.FILL_HORIZONTAL);
            mReportDir.setLayoutData(data2);

            data2 = new GridData(GridData.FILL_HORIZONTAL);
            mConnURL.setLayoutData(data2);
      }

      /**
       * Validates user input
       * @return true if required data present, false otherwise
       */
      private boolean validate() {
            boolean returnVal = false;
            if (mName.getText() == null || mName.getText().equals(""))
{
                  setErrorMessage("Name value must not be empty.");
            } else if (!mIsFolder) {
                  if (mHome.getText() == null ||
mHome.getText().equals("")
                              || mWorkspace.getText() == null
                              || mWorkspace.getText().equals("")
                              || mProject.getText() == null
                              || mProject.getText().equals("")
                              || mTestSuite.getText() == null
```

```
                              || mTestSuite.getText().equals("")
                              || mReportDir.getText() == null
                              || mReportDir.getText().equals("")
                              || mConnURL.getText() == null
                              || mConnURL.getText().equals("")) {
                        setErrorMessage("JUnit properties must not be
empty.");
                  } else {
                        returnVal = true;
                  }
            } else {
                  setErrorMessage(null);
                  returnVal = true;
            }
            return returnVal;
      }

      /**
       * Replace the OK button by createButton() from Dialog
       * @see org.eclipse.jface.dialogs.Dialog#
       *
createButtonsForButtonBar(org.eclipse.swt.widgets.Composite)
       */
      protected void createButtonsForButtonBar(Composite parent) {
            // Create Ok button
            mOKBtn = createButton(parent, OK, "Ok", true);
            mOKBtn.setEnabled(true);
            mOKBtn.addSelectionListener(new SelectionAdapter() {
                  public void widgetSelected(SelectionEvent e) {
                        if (validate()) {
                              mTestStub = new Test();
                              mTestStub.setName(mName.getText());

      mTestStub.setDescription(mDescription.getText());
                              mTestStub.setIsFolder('N');
                              Test parent = new Test();
                              parent.setId(mParentID);
                              parent.addChild(mTestStub);
                              mTestStub.setParent(parent);
                              TestType testType = new TestType();
                              testType.setName("JUnit");
                              mTestStub.setTestType(testType);
                              JunitTest junit = new JunitTest();
                              junit.setEclipseHome(mHome.getText());
                              junit.setWorkspace(mWorkspace.getText());
                              junit.setProject(mProject.getText());
                              junit.setTestSuite(mTestSuite.getText());
                              junit.setReportDir(mReportDir.getText());

      junit.setTptpConnection(mConnURL.getText());
                              mTestStub.addJunitTest(junit);

                              setReturnCode(OK);
                              close();
                        }
                  }
            });
```

```
            // Create Cancel button
            Button cancelButton = createButton(parent, CANCEL,
"Cancel", false);
            // Add a SelectionListener
            cancelButton.addSelectionListener(new SelectionAdapter() {
                  public void widgetSelected(SelectionEvent e) {
                        setReturnCode(CANCEL);
                        close();
                  }
            });
      }

      /**
       * Getter
       * @return the test stub
       */
      public Test getTestStub() {
            return mTestStub;
      }

}
```

UpdateDialog.java

```
/*****************************************************************
 *
 * File     :     UpdateDialog.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a dialog for users to enter data for the
 *                update of a test folder or test definition
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs;

import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.dialogs.TitleAreaDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;

/*****************************************************************
 * File           :     UpdateDialog.java
 *
 * Description    :     Provides a dialog for users to enter data for
 *                      the update of a test folder or test definition
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c)2007 Bob Brady
 *****************************************************************/
public class UpdateDialog extends TitleAreaDialog implements Observer {
      /** ID for OK, large integer used to avoid system conflicts */
      public static final int OK = 9999;
      /** dialog OK button */
      private Button mOKBtn;
      /** Test GUI type */
      private boolean mIsFolder;
      /** Name of the test definition */
      private Text mName;
```

```
	/** Description of the test definition */
	private Text mDescription;
	/** Home directory of the test */
	private Text mHome;
	/** Workspace directory location */
	private Text mWorkspace;
	/** Test project directory */
	private Text mProject;
	/** Test suite directory */
	private Text mTestSuite;
	/** Test report directory */
	private Text mReportDir;
	/** TPTP connection URL to be used for execution */
	private Text mConnURL;
	/** Stubbed version of Test object */
	private Test mTestStub;
	/** TPTeam database ID of test */
	private int mTestID;

	/**
	 * Constructor for UpdateDialog.
	 *
	 * @param shell -
	 *            Containing shell
	 */
	public UpdateDialog(Shell shell, boolean isFolder) {
		super(shell);
		mIsFolder = isFolder;
	}

	/**
	 * Complete the dialog with
	 *      a title and a message
	 * @see org.eclipse.jface.window.Window#create()
	 */
	public void create() {
		super.create();
		setTitle("Update Test Node");
		setMessage("Once current values have been loaded, edit the
items to be changed.");
		if (mIsFolder)
			getShell().setSize(600, 250);
		else
			getShell().setSize(600, 400);

	Activator.getDefault().getEventAdminHandler().addObserver(this);
	}

	/**
	 * Fill center area of the dialog
	 * @see org.eclipse.jface.dialogs.Dialog#
	 *      createDialogArea(org.eclipse.swt.widgets.Composite)
	 */
	protected Control createDialogArea(Composite parent) {

		// Create new composite as container
		final Composite area = new Composite(parent, SWT.NULL);
```

```
            GridLayout gridLayout = new GridLayout();
            gridLayout.numColumns = 2;
            area.setLayout(gridLayout);

            GridData data = new GridData(GridData.FILL_HORIZONTAL);
            area.setLayoutData(data);

            Label nameLabel = new Label(area, SWT.NONE);
            nameLabel.setText("Name:");
            mName = new Text(area, SWT.BORDER);

            GridData nameLData = new GridData();
            nameLData.widthHint = 60;
            nameLabel.setLayoutData(nameLData);

            GridData nameData = new GridData(GridData.FILL_HORIZONTAL);
            mName.setLayoutData(nameData);

            Label descLabel = new Label(area, SWT.NONE);
            descLabel.setText("Description:");

            GridData descLData = new GridData();
            descLData.widthHint = 60;
            descLData.verticalSpan = 15;
            descLabel.setLayoutData(descLData);

            mDescription = new Text(area, SWT.MULTI | SWT.WRAP |
SWT.BORDER
                        | SWT.V_SCROLL);
            GridData descData = new GridData(GridData.FILL_BOTH);
            descData.verticalSpan = 15;
            mDescription.setLayoutData(descData);

            if (!mIsFolder) {
                  createTestArea(area);
            }

            return area;
      }

      /**
       * Helper function to create additional test
       * definition data inputs to dialog area
       *
       * @param area the dialog area
       */
      protected void createTestArea(Composite area) {
            Label homeLabel = new Label(area, SWT.NONE);
            homeLabel.setText("Eclipse Home:");
            mHome = new Text(area, SWT.BORDER);

            Label workLabel = new Label(area, SWT.NONE);
            workLabel.setText("Eclipse Workspace:");
            mWorkspace = new Text(area, SWT.BORDER);

            Label projLabel = new Label(area, SWT.NONE);
```

```
        projLabel.setText("Eclipse Project:");
        mProject = new Text(area, SWT.BORDER);

        Label testsuiteLabel = new Label(area, SWT.NONE);
        testsuiteLabel.setText("TPTP Testsuite:");
        mTestSuite = new Text(area, SWT.BORDER);

        Label reportLabel = new Label(area, SWT.NONE);
        reportLabel.setText("TPTP Report Dir:");
        mReportDir = new Text(area, SWT.BORDER);

        Label connLabel = new Label(area, SWT.NONE);
        connLabel.setText("TPTP Conn URL:");
        mConnURL = new Text(area, SWT.BORDER);

        GridData data = new GridData();
        data.widthHint = 100;
        homeLabel.setLayoutData(data);

        data = new GridData();
        data.widthHint = 100;
        projLabel.setLayoutData(data);

        data = new GridData();
        data.widthHint = 100;
        testsuiteLabel.setLayoutData(data);

        data = new GridData();
        data.widthHint = 100;
        reportLabel.setLayoutData(data);

        data = new GridData();
        data.widthHint = 100;
        connLabel.setLayoutData(data);

        GridData data2 = new GridData(GridData.FILL_HORIZONTAL);
        mHome.setLayoutData(data2);

        data2 = new GridData(GridData.FILL_HORIZONTAL);
        mWorkspace.setLayoutData(data2);

        data2 = new GridData(GridData.FILL_HORIZONTAL);
        mProject.setLayoutData(data2);

        data2 = new GridData(GridData.FILL_HORIZONTAL);
        mTestSuite.setLayoutData(data2);

        data2 = new GridData(GridData.FILL_HORIZONTAL);
        mReportDir.setLayoutData(data2);

        data2 = new GridData(GridData.FILL_HORIZONTAL);
        mConnURL.setLayoutData(data2);
    }

    /**
     * Validates user input
     * @return true if required data present, false otherwise
```

```
	 */
	private boolean validate() {
		boolean returnVal = false;
		if (mName.getText() == null || mName.getText().equals(""))
{
			setErrorMessage("Name value must not be empty.");
		} else if (!mIsFolder) {
			if (mHome.getText() == null ||
mHome.getText().equals("")
					|| mWorkspace.getText() == null
					|| mWorkspace.getText().equals("")
					|| mProject.getText() == null
					|| mProject.getText().equals("")
					|| mTestSuite.getText() == null
					|| mTestSuite.getText().equals("")
					|| mReportDir.getText() == null
					|| mReportDir.getText().equals("")
					|| mConnURL.getText() == null
					|| mConnURL.getText().equals("")) {
				setErrorMessage("JUnit properties must not be
empty.");
			} else {
				returnVal = true;
			}
		} else {
			setErrorMessage(null);
			returnVal = true;
		}
		return returnVal;
	}

	/**
	 * Replace the OK button by createButton() from Dialog
	 * @see org.eclipse.jface.dialogs.Dialog#
	 *
createButtonsForButtonBar(org.eclipse.swt.widgets.Composite)
	 */
	protected void createButtonsForButtonBar(Composite parent) {
		// Create Ok button
		mOKBtn = createButton(parent, OK, "Ok", true);
		// Initially deactivate it
		mOKBtn.setEnabled(false);
		// Add a SelectionListener
		mOKBtn.addSelectionListener(new SelectionAdapter() {
			public void widgetSelected(SelectionEvent e) {
				if (validate()) {
					mTestStub = new Test();
					mTestStub.setId(mTestID);
					mTestStub.setName(mName.getText());

	mTestStub.setDescription(mDescription.getText());
					if(!mIsFolder)
					{
						JunitTest junit = new JunitTest();

	junit.setEclipseHome(mHome.getText());
```

```
	junit.setWorkspace(mWorkspace.getText());

	junit.setProject(mProject.getText());

	junit.setTestSuite(mTestSuite.getText());

	junit.setReportDir(mReportDir.getText());

	junit.setTptpConnection(mConnURL.getText());
                                    mTestStub.addJunitTest(junit);
                              }
                              setReturnCode(OK);
                              close();
                        }
                  }
            });

            // Create Cancel button
            Button cancelButton = createButton(parent, CANCEL,
"Cancel", false);
            // Add a SelectionListener
            cancelButton.addSelectionListener(new SelectionAdapter() {
                  public void widgetSelected(SelectionEvent e) {
                        setReturnCode(CANCEL);
                        close();
                  }
            });
      }

      public void update(Observable observable, Object object) {
            if (observable instanceof EventAdminHandler
                        && object instanceof TPEvent) {

                  TPEvent tpEvent = (TPEvent) object;

                  if (tpEvent.getTopic()

      .equals(ITPBridge.TEST_UPDATE_DATA_RESP_TOPIC)) {
                        String testXML = tpEvent.getDictionary().get(
                                    TPEvent.TEST_XML_KEY);
                        Test testStub =
TestXML.getTestFromXML(testXML);
                        populateWidgets(testStub);
                  }
            }
      }

      private void populateWidgets(final Test test) {
            Display.getDefault().syncExec(new Runnable() {
                  public void run() {
                        mTestID = test.getId();
                        mName.setText(test.getName());
                        if (test.getDescription() != null)

      mDescription.setText(test.getDescription());
```

```
                        if (test.getJunitTests() != null
                                    && test.getJunitTests().size() > 0)
{
                              for (JunitTest junit :
test.getJunitTests()) {

      mHome.setText(junit.getEclipseHome());

      mWorkspace.setText(junit.getWorkspace());

      mProject.setText(junit.getProject());

      mTestSuite.setText(junit.getTestSuite());

      mReportDir.setText(junit.getReportDir());

      mConnURL.setText(junit.getTptpConnection());
                              }
                        }
                        mOKBtn.setEnabled(true);
                  }
            });

      }

      public boolean close() {

      Activator.getDefault().getEventAdminHandler().deleteObserver(this
);
            super.close();
            return true;
      }

      /**
       * Getter
       * @return the test stub
       */
      public Test getTestStub()
      {
            return mTestStub;
      }

}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin

EventAdminClient.java

```
/**********************************************************************
 *
 * File     :     EventAdminClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A client of the EventAdmin Service
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin;

import java.util.Hashtable;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventAdmin;
import org.osgi.util.tracker.ServiceTracker;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/**********************************************************************
 * File           :     EventAdminClient.java
 *
 * Description    :     A client of the EventAdmin Service.  It sends
 *                      TPTeam Event data encapsulated as an OSGi Event
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class EventAdminClient {
      /** The EventAdmin ServiceTracker */
      private ServiceTracker mServiceTracker;

      /**
       * Constructor, gets and opens a tracker for
       * the EventAdmin OSGi Service
       *
       * @param context The TPBridge plug-in context
       */
      public EventAdminClient(BundleContext context) {
            mServiceTracker = new ServiceTracker(context,
EventAdmin.class
                        .getName(), null);
            mServiceTracker.open();
      }

      /**
       * Sends an OSGi Event to the OSGi EventAdmin Service
       * @param topic the event topic String
       * @param dictionary a table of key=value pairs
       * @return true if message sent successfully, false otherwise
```

```
       */
      public boolean sendEvent(String topic, Hashtable<String, String>
dictionary) {
            boolean messageSent = false;

            System.out.println("TPBuddy EventAdminClient SendEvent");
            EventAdmin eventAdmin = (EventAdmin)
mServiceTracker.getService();

            if (eventAdmin != null) {
                        dictionary.put(ITPBridge.SHARED_OBJECT_ID_KEY,
ITPBridge.DEFAULT_SHARED_OBJECT_ID);
                        System.out.println("EventAdminClient: Sent " +
topic + " Event for " + dictionary.get(TPEvent.TEST_NAME_KEY) +
                              " with containerID " +
dictionary.get(ITPBridge.CONTAINER_ID_KEY) +
                              " and soID " +
dictionary.get(ITPBridge.SHARED_OBJECT_ID_KEY));
                        eventAdmin.sendEvent(new Event(topic,
dictionary));
                        messageSent = true;
            }
            else
            {
                  System.out.println("TPBuddy EventAdminClient:
eventAdmin is null");
            }
            return messageSent;
      }
}
```

EventAdminHandler.java

```
/*********************************************************************
 *
 * File     :     EventAdminHandler.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The OSGi Service that handles OSGi Events from the
 *                EventAdmin Service and acts as an observable of
 *                TPTeam Events will notify various TPBuddy views
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Observable;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventConstants;
import org.osgi.service.event.EventHandler;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*********************************************************************
 * File           :     EventAdminHandler.java
 *
 * Description    :     The OSGi Service that handles OSGi Events from
 *                      the EventAdmin Service and acts as an
 *                      observable of TPTeam Events will notify various
 *                      TPBuddy views
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class EventAdminHandler extends Observable implements
EventHandler {
      /** Table of key=value pairs of the OSGi Event */
      private Hashtable<String, String[]> mDictionary = new
Hashtable<String, String[]>();
      /** A List log of all TPEvents received */
      private ArrayList<Event> mEvents;

      /**
       * Constructor
       *
       * Sets the OSGi Event topics to be handled by this service
       *
       * Sets the
       * @param context
       */
      public EventAdminHandler(BundleContext context) {
            mEvents = new ArrayList<Event>();
            mDictionary.put(EventConstants.EVENT_TOPIC,
```

```
                        new String[] {ITPBridge.TEST_EXEC_REQ_TOPIC,
ITPBridge.TEST_EXEC_RESULT_TOPIC,
                        ITPBridge.PROJ_GET_REQ_TOPIC,
ITPBridge.PROJ_GET_RESP_TOPIC,
                        ITPBridge.TEST_TREE_GET_REQ_TOPIC,
ITPBridge.TEST_TREE_GET_RESP_TOPIC,
                        ITPBridge.TEST_DEL_REQ_TOPIC,
ITPBridge.TEST_DEL_RESP_TOPIC,
                        ITPBridge.TEST_DETAIL_REQ_TOPIC,
ITPBridge.TEST_DETAIL_RESP_TOPIC,
                        ITPBridge.TEST_UPDATE_DATA_REQ_TOPIC,
ITPBridge.TEST_UPDATE_DATA_RESP_TOPIC,
                        ITPBridge.TEST_UPDATE_REQ_TOPIC,
ITPBridge.TEST_UPDATE_RESP_TOPIC,
                        ITPBridge.TEST_ADD_REQ_TOPIC,
ITPBridge.TEST_ADD_RESP_TOPIC,
                        ITPBridge.CHART_GET_DATA_REQ_TOPIC,
ITPBridge.CHART_GET_DATA_RESP_TOPIC});

            context.registerService(EventHandler.class.getName(), this,
mDictionary);
      }

      /**
       * Handles an OSGi Event from the EventAdmin Service
       * and notifies all registered TPBuddy Views so that
       * they perform any necessary updates.
       *
       * @param event the OSGi Event to be handled
       */
      public void handleEvent(Event event) {
            System.out.println("TPBuddy EventAdminHandler: Got " +
event.getTopic() + " Event for " +
event.getProperty(TPEvent.TEST_NAME_KEY));
            TPEvent tpEvent = new TPEvent(event);
            setChanged();
            notifyObservers(tpEvent);
      }

      /**
       * Getter
       * @return the List log of TPTeam Events
       */
      public ArrayList<Event> getEventLog() {
            return mEvents;
      }

}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.tpbridge

TPBridgeClient.java

```
/*****************************************************************
 *
 * File     :     TPBridgeClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Convenience client for the TPBridge OSGi Service
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.tpbridge;

import java.util.Hashtable;

import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.security.IConnectContext;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.ui.PlatformUI;
import org.osgi.framework.BundleContext;

import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.Client;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;

/*****************************************************************
 * File           :     TPBridgeClient.java
 *
 * Description    :     Convenience client for the TPBridge OSGi
 *                      Service
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TPBridgeClient extends Client {

      /**
       * Constructor
       *
       * @param context
       *            the TPBuddy plug-in context
       */
      public TPBridgeClient(BundleContext context) {
            super(context);

      setTPMgrECFID(getTPTeamProps().getProperty(TPMANAGER_ECFID_KEY));
      }

      /**
```

```
       * Connects the ECF container to the communication server,
Associates the
       * connected container with the TPBridge, Sends a project request
message to
       * TPManager
       *
       * @param container
       *            the ECF XMPPS Container
       * @param targetID
       *            the ECF ID of the client
       * @param connectContext
       *            the ECF connection context
       */
      public void connect(IContainer container, ID targetID,
                  IConnectContext connectContext) {
            try {
                  container.connect(targetID, connectContext);
                  setContainer(container, targetID.getName(),
ITPBridge.TPTEAM_BUDDY);
                  sendProjGetRequest(targetID.getName());
            } catch (final Exception e) {
                  MessageDialog.openError(PlatformUI.getWorkbench()
                              .getActiveWorkbenchWindow().getShell(),
                              "TPBuddy Login Error",
"ConnectionException: "
                                          + e.getMessage());

            }
      }

      /**
       * Sends a project request message to TPManager
       *
       * @param ecfID
       *            the ECF ID of the intended recipient
       */
      private void sendProjGetRequest(String ecfID) {
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO, getTPMgrECFID());
            dictionary.put(TPEvent.FROM, ecfID);
            if (Activator.getDefault() != null)

      Activator.getDefault().getEventAdminClient().sendEvent(
                              ITPBridge.PROJ_GET_REQ_TOPIC,
dictionary);
      }

}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.views

DetailLabelProvider.java

```
/*********************************************************************
 *
 * File     :     DetailLabelProvider.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the text and images for the columns in the
 *                Test Details View table
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.HashMap;
import java.util.Iterator;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.jface.viewers.ITableLabelProvider;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;
import org.eclipse.ui.ISharedImages;
import org.eclipse.ui.PlatformUI;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;

/*********************************************************************
 * File           :     DetailLabelProvider.java
 *
 * Description    :     Provides the text and images for the columns in
 *                      the Test Details View table
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class DetailLabelProvider extends LabelProvider implements
            ITableLabelProvider {

      private HashMap<ImageDescriptor, Image> imageCache = new
HashMap<ImageDescriptor, Image>();

      /**
       * Removes images from the cache
       */
      @SuppressWarnings("unchecked")
      public void dispose() {
            for (Iterator i = imageCache.values().iterator();
i.hasNext();) {
                  ((Image) i.next()).dispose();
            }
            imageCache.clear();
      }
```

```
	/**
	 * Extracts table column text
	 * @param element the Object to extract text from
	 * @param index the column index
	 */
	public String getColumnText(Object element, int index) {
		TPEntity tpEntity = (TPEntity) element;
		switch (index) {
		case 0:
			return tpEntity.getType();
		case 1:
			return tpEntity.getDescription();
		default:
			return "unknown " + index;
		}
	}

	/**
	 * Gets the appropriate column image
	 *
	 * @param element the object to get column image
	 * @param columnIndex the column index
	 */
	public Image getColumnImage(Object element, int columnIndex) {
		TPEntity tpEntity = (TPEntity) element;
		Image image = null;
		if (tpEntity.getDescription() == null || columnIndex == 1)
			return image;

		if (tpEntity.getDescription().equals(TPEntity.FOLDER)) {
			image =
PlatformUI.getWorkbench().getSharedImages().getImage(
					ISharedImages.IMG_OBJ_FOLDER);
		} else if
(tpEntity.getDescription().equals(TPEntity.JUNIT_TEST)) {
			image = getImageFromString("icons/junit.gif");
		} else if (tpEntity.getType().equals(TPEntity.EXEC_PASS)) {
			image = getImageFromString("icons/testok.gif");
		} else if (tpEntity.getType().equals(TPEntity.EXEC_FAIL)) {
			image = getImageFromString("icons/testfail.gif");
		} else if (tpEntity.getType().equals(TPEntity.EXEC_ERROR))
{
			image = getImageFromString("icons/testerr.gif");
		} else if
(tpEntity.getType().equals(TPEntity.EXEC_INCONCLUSIVE)) {
			image = getImageFromString("icons/testignored.gif");
		} else if
(tpEntity.getType().toUpperCase().indexOf("ECLIPSE") >= 0) {
			image = getImageFromString("icons/eclipse.gif");
		} else if (tpEntity.getType().toUpperCase().indexOf("TPTP")
>= 0) {
			image = getImageFromString("icons/eclipse.gif");
		}

		return image;
	}
```

```
	/**
	 * Helper method to get an image from a String
	 * type descriptor
	 *
	 * @param type String type of image
	 * @return the image
	 */
	private Image getImageFromString(String type) {
		ImageDescriptor descriptor =
Activator.getImageDescriptor(type);
		// obtain the cached image corresponding to the descriptor
		Image image = (Image) imageCache.get(descriptor);
		if (image == null) {
			image = descriptor.createImage();
			imageCache.put(descriptor, image);
		}
		return image;
	}

}
```

DetailView.java

```
/*****************************************************************
 *
 * File     :     DetailView.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the view of a Test's details
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.viewers.ArrayContentProvider;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.ui.part.ViewPart;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;

/*****************************************************************
 * File           :     DetailView.java
 *
 * Description    :     Provides the view of a Test's details
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class DetailView extends ViewPart implements Observer {
      /** The view ID */
      public static final String ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.views.detailview";
      /** The encapsulated table for rendering test details */
      private Table mTable;
      /** The TableViewer for rendering test details */
      private TableViewer mTableViewer;

      /**
       * Constructor
       */
      public DetailView() {
      }

      /**
       * Initializes the view by adding it as an observer to the
```

```
	 * TPBuddy EventAdminHandler and creating its TableViewer
	 * @param parent the GUI parent to this view
	 */
	@Override
	public void createPartControl(Composite parent) {

	Activator.getDefault().getEventAdminHandler().addObserver(this);
		initTableViewer(parent);
	}

	@Override
	public void setFocus() {

	}

	/**
	 * Initializes the TableView with label and content
	 * providers
	 * @param parent the Composite parent to the view
	 */
	private void initTableViewer(Composite parent) {
		mTableViewer = new TableViewer(parent, SWT.SINGLE |
SWT.FULL_SELECTION);
		initTable();
		initColumns();
		mTableViewer.setLabelProvider(new DetailLabelProvider());
		mTableViewer.setContentProvider(new
ArrayContentProvider());
	}

	/**
	 * Helper method to set header & lines visible
	 */
	private void initTable() {
		mTable = mTableViewer.getTable();
		mTable.setHeaderVisible(true);
		mTable.setLinesVisible(true);
	}

	/**
	 * Helper method to set column properties
	 */
	private void initColumns() {
		String[] columnNames = new String[] { "Property", "Value"
};
		int[] columnWidths = new int[] { 150, 500 };
		int[] columnAlignments = new int[] { SWT.LEFT, SWT.LEFT };
		for (int i = 0; i < columnNames.length; i++) {
			TableColumn tableColumn = new TableColumn(mTable,
					columnAlignments[i]);
			tableColumn.setText(columnNames[i]);
			tableColumn.setWidth(columnWidths[i]);
		}
	}

	/**
	 * Update called when TPBuddy EventAdminHandler receives
```

```
       * a test detail response TPEvent
       *
       * If the event is a test details response, then this view
       * will extract the TPEntity and render its details
       *
       * @param observable the object that called the update
       * @param object the TPEvent to be handled
       */
      public void update(Observable observable, Object object) {
            if (observable instanceof EventAdminHandler
                        && object instanceof TPEvent) {
                  TPEvent tpEvent = (TPEvent) object;
                  System.out.println("DetailView Got Update: " +
tpEvent.getTopic());
                  if
(tpEvent.getTopic().equals(ITPBridge.TEST_DETAIL_RESP_TOPIC)) {

                        System.out.println("DetailView: update called
for "
                                    + tpEvent.getTopic() + " Event for
"
                                    + tpEvent.getID());

                        String testPropXML =
tpEvent.getDictionary().get(
                                    TPEvent.TEST_PROP_XML_KEY);

                        System.out.println(testPropXML);

                        final TPEntity[] tpEntities =
TestXML.getTPEntitiesFromXML(testPropXML);

                        Display.getDefault().syncExec(new Runnable() {
                              public void run() {
                                    mTableViewer.setInput(tpEntities);
                              }
                        });
                  }
            }
      }
}
```

EventHistoryView.java

```
/*****************************************************************
 *
 * File     :     EventHistoryView.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a view of all TPEvents received by TPBuddy
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.ArrayList;
import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.viewers.ArrayContentProvider;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.ui.part.ViewPart;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.tpbridge.TPBridgeClient;

/*****************************************************************
 * File           :     EventHistoryView.java
 *
 * Description    :     Provides a view of all TPEvents received by
 *                      TPBuddy
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class EventHistoryView extends ViewPart implements Observer {
      /** The view ID */
      public static final String ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.eventhistview";
      /** The encapsulated SWT Table used */
      private Table mTable;
      /** The TableViewer that renders the TPEvents */
      private TableViewer mTableViewer;
      /** Helper class for updating view */
      private TableUpdater mTableUpdater;
      /** Reference to the TPBuddy TPBridge client */
      private TPBridgeClient mTPBridgeClient;

      /**
       * Constructor
       *
       * Adds this view as an observer to the
```

```
	 * TPBuddy EventAdminHandler and gets a handle to
	 * the TPBuddy TPBridge client
	 */
	public EventHistoryView() {
		mTPBridgeClient =
Activator.getDefault().getTPBridgeClient();

	Activator.getDefault().getEventAdminHandler().addObserver(this);
	}

	@Override
	public void createPartControl(Composite parent) {
		initTableViewer(parent);
		mTableUpdater = new TableUpdater(mTableViewer);
	}

	/**
	 * Initializes the TableViewer
	 * @param parent the Composite parent of the view
	 */
	private void initTableViewer(Composite parent) {
		mTableViewer = new TableViewer(parent, SWT.SINGLE |
SWT.FULL_SELECTION);
		initTable();
		initColumns();
		mTableViewer.setLabelProvider(new TPEventLabelProvider());
		mTableViewer.setContentProvider(new
ArrayContentProvider());
		mTableViewer.setInput(getTPBridgeEvents());
	}

	private void initTable() {
		mTable = mTableViewer.getTable();
		mTable.setHeaderVisible(true);
		mTable.setLinesVisible(true);
	}

	/**
	 * Initializes column display properties
	 */
	private void initColumns() {
		String[] columnNames = new String[] { "From", "To",
"Topic", "Project", "Test", "ID", "Status" };
		int[] columnWidths = new int[] { 150, 150, 150, 150, 150,
150, 150 };
		int[] columnAlignments = new int[] { SWT.LEFT, SWT.LEFT,
SWT.LEFT, SWT.LEFT,
				SWT.LEFT, SWT.LEFT, SWT.LEFT };
		for (int i = 0; i < columnNames.length; i++) {
		  TableColumn tableColumn = new TableColumn(mTable,
				columnAlignments[i]);
		  tableColumn.setText(columnNames[i]);
		  tableColumn.setWidth(columnWidths[i]);
		}

	}
```

```
	@Override
	public void setFocus() {
	}

	/**
	 * Helper class
	 *
	 * Adds TPEvents to the view's table as they
	 * are received in real time.
	 *
	 */
	private static class TableUpdater implements Runnable {

		private Object mTableViewerObject = null;

		private TableViewer mTableViewer;

		public TableUpdater(TableViewer tableViewer) {
			mTableViewer = tableViewer;
		}

		public void insertObject(Object objectToInsert) {
			mTableViewerObject = objectToInsert;
			Display.getDefault().syncExec(this);
		}

		public void run() {
			mTableViewer.add(mTableViewerObject);
		}
	}

	private ArrayList<TPEvent> getTPBridgeEvents() {
		return mTPBridgeClient.getEventLog();
	}

	public void dispose() {
		super.dispose();

	Activator.getDefault().getEventAdminHandler().deleteObserver(this
);
	}

	/**
	 * Update called when TPBuddy EventAdminHandler receives
	 * a TPTeam event.  All received TPEvents will be added
	 * to the view's table
	 *
	 * @param observable the object that called the update
	 * @param object the TPEvent to be handled
	 */
	public void update(Observable observable, Object object) {
		if (observable instanceof EventAdminHandler && object
instanceof TPEvent) {
			TPEvent tpEvent = (TPEvent) object;
			System.out.println("EventHistoryView: update called
for "
					+ tpEvent.getTopic() + " Event for "
```

```
                                + tpEvent.getTestName());
                        mTableUpdater.insertObject(tpEvent);
                    }
                }
            }
```

ProjectLabelProvider.java

```
/*****************************************************************
 *
 * File     :     ProjectLabelProvider.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the text and images for the columns in the
 *                Project View table
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import org.eclipse.jface.viewers.ITableLabelProvider;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;

/*****************************************************************
 * File           :     ProjectLabelProvider.java
 *
 * Description    :     Provides the text and images for the columns in
 *                      the Project View table
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class ProjectLabelProvider extends LabelProvider implements
            ITableLabelProvider {

      /**
       * Extracts table column text
       * @param element the Object to extract text from
       * @param index the column index
       */
      public String getColumnText(Object element, int index) {
            Project proj = (Project) element;
            switch (index) {
            case 0:
                  return String.valueOf(proj.getId());
            case 1:
                  return proj.getName();
            case 2:
                  return proj.getDescription();
            case 3:
                  return proj.getProduct().getName();
            default:
                  return "unknown " + index;
            }
      }

      /**
       * Gets the appropriate column image
       *
       * @param element the object to get column image
```

```
         * @param columnIndex the column index
         */
        public Image getColumnImage(Object element, int columnIndex) {
              return null;
        }
}
```

ProjectView.java

```
/*****************************************************************
 *
 * File     :     ProjectView.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a view of the TPTeam projects available
 *                to a user
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.Hashtable;
import java.util.List;
import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.IToolBarManager;
import org.eclipse.jface.viewers.ArrayContentProvider;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.viewers.SelectionChangedEvent;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.part.ViewPart;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.ProjectXML;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;

/*****************************************************************
 * File           :     ProjectView.java
 *
 * Description    :     Provides a view of the TPTeam projects
 *                      available to a user
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*****************************************************************/
public class ProjectView extends ViewPart implements Observer {
      /** The view ID */
```

```
	public static final String ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.projectview";
	/** The encapsulated SWT Table */
	private Table mTable;
	/** The TableView that encapsulates the SWT Table */
	private TableViewer mTableViewer;
	/** Action for requesting a project's test tree */
	private Action mGetTestTree;
	/** Action for requesting a project's report charts */
	private Action mGetProjReports;

	/**
	 * Constructor
	 *
	 * Adds this view as an observer to the TPBuddy
EventAdminHandler.
	 * Creates the test tree and report request Actions.
	 */
	public ProjectView() {

	Activator.getDefault().getEventAdminHandler().addObserver(this);
		mGetTestTree = new Action("Get Proj TestTree") {
			public void run() {
				getTestTreeAction();
			}
		};
		mGetProjReports = new Action("Get Proj Reports") {
			public void run() {
				getProjReportsAction();
			}
		};

	}

	/**
	 * Requests the test tree TPEntity from the TPManager for
	 * the selected project in the view.
	 */
	private void getTestTreeAction() {
		IStructuredSelection selection = (IStructuredSelection)
mTableViewer
				.getSelection();
			Project proj = (Project) selection.getFirstElement();
			Hashtable<String, String> dictionary = new
Hashtable<String, String>();
			dictionary.put(TPEvent.SEND_TO,
Activator.getDefault()
					.getTPBridgeClient().getTPMgrECFID());
			dictionary.put(TPEvent.FROM, Activator.getDefault()
					.getTPBridgeClient().getTargetIDName());
			dictionary
					.put(TPEvent.PROJECT_ID_KEY,
String.valueOf(proj.getId()));
			dictionary.put(TPEvent.PROJECT_KEY, proj.getName());
			System.out.println("Project Name: " +
dictionary.get(TPEvent.PROJECT_KEY));
			showTestView();
```

```
      Activator.getDefault().getEventAdminClient().sendEvent(
                                 ITPBridge.TEST_TREE_GET_REQ_TOPIC,
dictionary);

      }

      /**
       * Requests the pie chart TPEntity from the TPManager for
       * the selected project in the view.
       */

      private void getProjReportsAction() {
            IStructuredSelection selection = (IStructuredSelection)
mTableViewer
                        .getSelection();
                  Project proj = (Project) selection.getFirstElement();

                  Hashtable<String, String> dictionary = new
Hashtable<String, String>();
                  dictionary.put(TPEvent.SEND_TO,
Activator.getDefault()
                              .getTPBridgeClient().getTPMgrECFID());
                  dictionary.put(TPEvent.FROM, Activator.getDefault()
                              .getTPBridgeClient().getTargetIDName());
                  dictionary
                              .put(TPEvent.PROJECT_ID_KEY,
String.valueOf(proj.getId()));
                  dictionary.put(TPEvent.PROJECT_KEY, proj.getName());
                  dictionary.put(ChartDataSet.CHART_TYPE,
ChartDataSet.PIE);
                  showProjReportView();

      Activator.getDefault().getEventAdminClient().sendEvent(
                                 ITPBridge.CHART_GET_DATA_REQ_TOPIC,
dictionary);

      }


      @Override
      public void createPartControl(Composite parent) {
            initTableViewer(parent);
            createActions();
      }

      @Override
      public void setFocus() {

      }

      /**
       * Helper method that will give the test tree
       * view focus in the tabbed view of TPBuddy
       */
      private void showTestView() {
            IWorkbenchPage page = PlatformUI.getWorkbench()
```

```
                        .getActiveWorkbenchWindow().getActivePage();
            if (page != null) {
                  try {
                        page.showView(TestView.ID);
                  } catch (PartInitException e) {
                        e.printStackTrace();
                  }
            }
      }

      /**
       * Helper method that will give the report
       * view focus in the tabbed view of TPBuddy
       */
      private void showProjReportView() {
            IWorkbenchPage page = PlatformUI.getWorkbench()
                        .getActiveWorkbenchWindow().getActivePage();
            if (page != null) {
                  try {
                        page.showView(ReportView.ID);
                  } catch (PartInitException e) {

                        e.printStackTrace();
                  }
            }
      }

      /**
       * Helper class that initializes Actions
       */
      private void createActions() {
            mGetProjReports.setEnabled(true);
            mGetProjReports.setImageDescriptor(Activator
                        .getImageDescriptor("icons/report.gif"));
            mGetTestTree.setEnabled(true);
            mGetTestTree.setImageDescriptor(Activator
                        .getImageDescriptor("icons/testhier.gif"));
            mTableViewer
                        .addSelectionChangedListener(new
ISelectionChangedListener() {
                              public void
selectionChanged(SelectionChangedEvent event) {
                                    // updateAction();
                              }
                        });
            IToolBarManager mgr =
getViewSite().getActionBars().getToolBarManager();
            mgr.add(mGetProjReports);
            mgr.add(mGetTestTree);

      }

      private void initTableViewer(Composite parent) {
            mTableViewer = new TableViewer(parent, SWT.SINGLE |
SWT.FULL_SELECTION);
            initTable();
            initColumns();
```

```
            mTableViewer.setLabelProvider(new ProjectLabelProvider());
            mTableViewer.setContentProvider(new
ArrayContentProvider());
      }

      private void initTable() {
            mTable = mTableViewer.getTable();
            mTable.setHeaderVisible(true);
            mTable.setLinesVisible(true);
      }

      /**
       * Initializes the TableView column sizes, headers,
       * and other properties
       */
      private void initColumns() {
            String[] columnNames = new String[] { "ID", "Name",
"Description",
                        "Product" };
            int[] columnWidths = new int[] { 50, 150, 300, 150 };
            int[] columnAlignments = new int[] { SWT.LEFT, SWT.LEFT,
SWT.LEFT,
                        SWT.LEFT };
            for (int i = 0; i < columnNames.length; i++) {
                  TableColumn tableColumn = new TableColumn(mTable,
                              columnAlignments[i]);
                  tableColumn.setText(columnNames[i]);
                  tableColumn.setWidth(columnWidths[i]);
            }

      }

      public void dispose() {
            super.dispose();

      Activator.getDefault().getEventAdminHandler().deleteObserver(this
);
      }

      /**
       * Update used to refresh view when a project/product get
       * response is received
       *
       * @param observable the object issuing the update
       * @param object the TPEvent to be handled
       */
      public void update(Observable observable, Object object) {
            if (observable instanceof EventAdminHandler
                        && object instanceof TPEvent
                        && ((TPEvent) object).getTopic().equals(
                                    ITPBridge.PROJ_GET_RESP_TOPIC)) {
                  TPEvent tpEvent = (TPEvent) object;
                  System.out.println("ProjectView: update called for "
                              + tpEvent.getTopic());
                  final List<Project> projs =
ProjectXML.getProjsFromXML(tpEvent
```

```
      .getDictionary().get(TPEvent.PROJ_PROD_XML_KEY));
                  System.out.println(tpEvent

      .getDictionary().get(TPEvent.PROJ_PROD_XML_KEY));
                  for(Project proj : projs)
                  {
                        System.out.println("Proj Name: " +
proj.getName() + ", Desc: " + proj.getDescription());
                  }
                  Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                              mTableViewer.setInput(projs);
                        }
                  });
            }
      }
}
```

ReportView.java

```
/*****************************************************************
 *
 * File     :     ReportView.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a view of the various project report charts
 *                available from TPTeam
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.Hashtable;
import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.IToolBarManager;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.part.ViewPart;
import org.jfree.chart.JFreeChart;
import org.jfree.experimental.chart.swt.ChartComposite;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.ChartDataSetXML;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.charts.BarChart;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.charts.LineChart;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.charts.PieChart;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;

/*****************************************************************
 * File           :     ReportView.java
 *
 * Description    :     Provides a view of the various project report
 *                      charts available from TPTeam
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class ReportView extends ViewPart implements Observer {
      /** The view ID */
      public static final String ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.views.reportview";
      /** Action to send pie chart request */
      private Action mGetPieChart;
      /** Action to send bar chart request */
```

```
	private Action mGetBarChart;
	/** Action to send line chart request */
	private Action mGetLineChart;
	/** The GUI Composite parent to the view */
	private Composite mParent;
	/** A Composite frame to hold the view's chart */
	private ChartComposite mFrame;
	/** The JFreeChart that the view renders */
	private JFreeChart mChart;
	/** The TPTeam database ID of the project */
	private String mProjID;
	/** The TPTeam name of the project */
	private String mProjName;

	/**
	 * Constructor, creates all chart request actions
	 */
	public ReportView() {
		mGetPieChart = new Action("Get Proj Overview Chart") {
			public void run() {
				getPieAction();
			}
		};
		mGetBarChart = new Action("Get Proj User Chart") {
			public void run() {
				getBarAction();
			}
		};
		mGetLineChart = new Action("Get Proj Time Chart") {
			public void run() {
				getLineAction();
			}
		};
	}

	/**
	 * Sends a chart request event to the TPManager,
	 * of type pie chart
	 */
	private void getPieAction() {
		Shell parent = getViewSite().getShell();
		if (mProjID == null || mProjID.equalsIgnoreCase("")
				|| Integer.parseInt(mProjID) < 1) {
			MessageDialog
					.openError(
							parent,
							"Report View Error",
							"A Project must first Be
selected in the Project View, then click on the \"Get Proj Reports\"
icon.");
			return;
		}

		Hashtable<String, String> dictionary = new
Hashtable<String, String>();
		dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
				.getTPBridgeClient().getTPMgrECFID());
```

```
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
            dictionary.put(TPEvent.PROJECT_KEY, mProjName);
            dictionary.put(ChartDataSet.CHART_TYPE, ChartDataSet.PIE);

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.CHART_GET_DATA_REQ_TOPIC,
dictionary);
      }

      /**
       * Sends a chart request event to the TPManager,
       * of type bar chart
       */

      private void getBarAction() {
            Shell parent = getViewSite().getShell();
            if (mProjID == null || mProjID.equalsIgnoreCase("")
                        || Integer.parseInt(mProjID) < 1) {
                  MessageDialog
                              .openError(
                                          parent,
                                          "Report View Error",
                                          "A Project must first Be
selected in the Project View, then click on the \"Get Proj Reports\"
icon.");
                  return;
            }

            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
            dictionary.put(TPEvent.PROJECT_KEY, mProjName);
            dictionary.put(ChartDataSet.CHART_TYPE, ChartDataSet.BAR);

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.CHART_GET_DATA_REQ_TOPIC,
dictionary);
      }

      /**
       * Sends a chart request event to the TPManager,
       * of type line chart
       */

      private void getLineAction() {
            Shell parent = getViewSite().getShell();
            if (mProjID == null || mProjID.equalsIgnoreCase("")
                        || Integer.parseInt(mProjID) < 1) {
                  MessageDialog
```

```
                              .openError(
                                          parent,
                                          "Report View Error",
                                          "A Project must first Be
selected in the Project View, then click on the \"Get Proj Reports\"
icon.");
                  return;
            }
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
            dictionary.put(TPEvent.PROJECT_KEY, mProjName);
            dictionary.put(ChartDataSet.CHART_TYPE, ChartDataSet.LINE);

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.CHART_GET_DATA_REQ_TOPIC,
dictionary);

      }

      /**
       * Gets the project ID and same of the currently
       * selected project in the ProjectView
       *
       * Sends out a get pie chart request to the TPManager
       *
       * @param parent the GUI Composite parent to the view
       */
      @Override
      public void createPartControl(Composite parent) {

      Activator.getDefault().getEventAdminHandler().addObserver(this);
            mParent = parent;
            GridLayout layout = new GridLayout();
            layout.numColumns = 1;
            mParent.setLayout(layout);
            GridData data = new GridData(GridData.FILL_BOTH);
            mParent.setLayoutData(data);

            mProjID = Activator.getDefault().getProjID();
            mProjName = Activator.getDefault().getProjName();

            if (mProjID != null && !mProjID.equalsIgnoreCase("")
                        && Integer.parseInt(mProjID) > 0)
                  getPieAction();

            createActions();
      }

      /**
       * Helper method to set various Action
       * properties
```

```
	 */
	private void createActions() {

		mGetPieChart.setEnabled(true);
		mGetPieChart.setImageDescriptor(Activator
				.getImageDescriptor("icons/piecharticon.gif"));

		mGetBarChart.setEnabled(true);
		mGetBarChart.setImageDescriptor(Activator

	.getImageDescriptor("icons/chartselector.gif"));

		mGetLineChart.setEnabled(true);
		mGetLineChart.setImageDescriptor(Activator

	.getImageDescriptor("icons/linecharticon.gif"));

		IToolBarManager mgr =
getViewSite().getActionBars().getToolBarManager();
		mgr.add(mGetPieChart);
		mgr.add(mGetBarChart);
		mgr.add(mGetLineChart);

	}

	/**
	 * Creates and shows the project bar chart of test executions
	 * vs. user who created the test definition
	 *
	 * @param dataSets the input ChartDataSet array
	 */
	private void createAndShowBarChart(ChartDataSet[] dataSets) {
		mChart = BarChart.getInstance().createChart(dataSets,
mProjName);
		mFrame.setChart(mChart);
		mFrame.layout();
	}

	/**
	 * Creates and shows the project line chart of project
	 * test execution status over time
	 *
	 * @param dataSets the input ChartDataSet array
	 */
	private void createAndShowLineChart(ChartDataSet dataSet) {
		mChart = LineChart.getInstance().createChart(new
ChartDataSet[]{dataSet}, mProjName);
		mFrame.setChart(mChart);
		mFrame.layout();
	}

	/**
	 * Creates and shows the project pie chart of all project
	 * test execution verdicts
	 *
	 * @param dataSets the input ChartDataSet array
	 */
```

```
	private void createAndShowPieChart(ChartDataSet dataSet) {
		mChart = PieChart.getInstance().createChart(new
ChartDataSet[]{dataSet}, mProjName);

		if (mFrame == null) {
			mFrame = new ChartComposite(mParent, SWT.NONE,
mChart, true);

			GridData data = new GridData(GridData.FILL_BOTH);
			mFrame.setLayoutData(data);
			mFrame.pack(true);
		} else {
			mFrame.setChart(mChart);
			mFrame.layout();
		}
	}

	@Override
	public void setFocus() {
	}

	/**
	 * Update used to refresh view when a get chart
	 * response TPEvent is received
	 *
	 * @param observable the object issuing the update
	 * @param object the TPEvent to be handled
	 */
	public void update(Observable observable, Object object) {

		if (observable instanceof EventAdminHandler
				&& object instanceof TPEvent) {

			// A Specific Chart Response was Received
			if (((TPEvent) object).getTopic().equals(
					ITPBridge.CHART_GET_DATA_RESP_TOPIC)) {
				TPEvent tpEvent = (TPEvent) object;

				// Extract the data set & chart type
				String dataSetXML =
tpEvent.getDictionary().get(
						TPEvent.CHART_DATASET_XML_KEY);
				String chartType = tpEvent.getDictionary().get(
						ChartDataSet.CHART_TYPE);

				// Handle pie chart responses
				if
(chartType.equalsIgnoreCase(ChartDataSet.PIE)) {
					final ChartDataSet dataSet =
ChartDataSetXML

	.getDataSetFromXML(dataSetXML);
					Display.getDefault().syncExec(new
Runnable() {
						public void run() {

	createAndShowPieChart(dataSet);
```

```
                                        mParent.layout();
                                    }
                                });

                            // Handle bar chart responses
                            } else if
(chartType.equalsIgnoreCase(ChartDataSet.BAR)) {
                                final ChartDataSet[] dataSets =
ChartDataSetXML

      .getDataSetsFromXML(dataSetXML);
                                Display.getDefault().syncExec(new
Runnable() {
                                    public void run() {

      createAndShowBarChart(dataSets);
                                        mParent.layout();
                                    }
                                });

                            // Handle line chart responses
                            } else if
(chartType.equalsIgnoreCase(ChartDataSet.LINE)) {
                                final ChartDataSet dataSet =
ChartDataSetXML

      .getDataSetFromXML(dataSetXML);
                                Display.getDefault().syncExec(new
Runnable() {
                                    public void run() {

      createAndShowLineChart(dataSet);
                                        mParent.layout();
                                    }
                                });
                            }

                    // Chart request went out, grab selected project
details now
                    } else if (((TPEvent) object).getTopic().equals(
                                ITPBridge.CHART_GET_DATA_REQ_TOPIC)) {
                            TPEvent tpEvent = (TPEvent) object;
                            mProjID =
tpEvent.getDictionary().get(TPEvent.PROJECT_ID_KEY);
                            mProjName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);
                            Activator.getDefault().setProjID(mProjID);
                            Activator.getDefault().setProjName(mProjName);

                    // Test tree request went out, grab selected project
details now
                    } else if (((TPEvent) object).getTopic().equals(
                                ITPBridge.TEST_TREE_GET_REQ_TOPIC)) {
                            TPEvent tpEvent = (TPEvent) object;
                            mProjID =
tpEvent.getDictionary().get(TPEvent.PROJECT_ID_KEY);
```

```
                        mProjName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);
                        Activator.getDefault().setProjID(mProjID);
                        Activator.getDefault().setProjName(mProjName);
                    }
                }
            }
        }
```

TestContentProvider.java

```
/*********************************************************************
 *
 * File     :     TestContentProvider.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the content for the Test View
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import org.eclipse.jface.viewers.ArrayContentProvider;
import org.eclipse.jface.viewers.ITreeContentProvider;
import org.eclipse.jface.viewers.TreeViewer;
import org.eclipse.jface.viewers.Viewer;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNode;
import
edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNodeChangeListener;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;

/*********************************************************************
 * File           :     TestContentProvider.java
 *
 * Description    :     Provides the content for the Test View
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class TestContentProvider extends ArrayContentProvider
implements
            ITreeContentProvider, ITreeNodeChangeListener {

      /** The Test View TreeViewer */
      private TreeViewer mTreeViewer;

      /**
       * Refreshes the tree when input has changed
       *
       * @param viewer
       *            the TestView's TreeViewer
       * @param oldInput
       *            the old data input
       * @param newIput
       *            the new data input
       */
      public void inputChanged(Viewer viewer, Object oldInput, Object
newInput) {
            mTreeViewer = (TreeViewer) viewer;
            if (oldInput != null) {
                  for (ITreeNode node : (TPEntity[]) oldInput)
                        removeListenerFrom(node);
            }
            if (newInput != null) {
```

```
                  for (ITreeNode node : (TPEntity[]) newInput)
                        addListenerTo(node);
            }
      }

      /**
       * Because the domain model does not have a richer listener
model,
       * recursively remove this listener from each child node of the
given
       * ITreeNode.
       *
       * @param node
       *            the ITreeNode
       */
      protected void removeListenerFrom(ITreeNode node) {
            node.removeChangeListener(this);
            for (ITreeNode child : node.getChildren()) {
                  removeListenerFrom(child);
            }
      }

      /**
       * Because the domain model does not have a richer listener
model,
       * recursively add this listener to each child node of the given
       * ITreeNode.
       *
       * @param node the ITreeNode
       */
      protected void addListenerTo(ITreeNode node) {
            node.addChangeListener(this);
            for (ITreeNode child : node.getChildren()) {
                  addListenerTo(child);
            }
      }

      // TreeViewer refresh operations in response to node CRUD

      public void updateNode(ITreeNode node) {
            mTreeViewer.refresh(node.getParent(), true);
      }

      public void addNode(ITreeNode node) {
            addListenerTo(node);
            mTreeViewer.refresh(node.getParent(), false);
      }

      public void deleteNode(ITreeNode node) {
            removeListenerFrom(node);
            mTreeViewer.refresh(node.getParent(), true);
      }


      // Public accessors

      public Object getParent(Object child) {
```

```
            return ((ITreeNode) child).getParent();
      }

      public Object[] getChildren(Object parent) {
            return ((ITreeNode) parent).getChildren().toArray(new
ITreeNode[0]);
      }

      public boolean hasChildren(Object parent) {
            return ((ITreeNode) parent).getChildren().size() > 0;
      }
}
```

TestLabelProvider.java

```
/*********************************************************************
 *
 * File     :     TestLabelProvider.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the text and images for the columns in the
 *                Test View tree
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.HashMap;
import java.util.Iterator;
import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;
import org.eclipse.ui.ISharedImages;
import org.eclipse.ui.PlatformUI;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;

/*********************************************************************
 * File           :     TestLabelProvider.java
 *
 * Description    :     Provides the text and images for the columns in
 *                      the Test View tree
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class TestLabelProvider extends LabelProvider {
      /** A cache for images used in the test tree */
      private HashMap<ImageDescriptor, Image> imageCache = new
HashMap<ImageDescriptor, Image>();

      /**
       * Clears the image cache
       */
      @SuppressWarnings("unchecked")
      public void dispose() {
            for (Iterator i = imageCache.values().iterator();
i.hasNext();) {
                  ((Image) i.next()).dispose();
            }
            imageCache.clear();
      }

      /**
       * Extracts the tree node name from
       * a given object
       *
       * @param obj the object
       */
```

```
	public String getText(Object obj) {
		if (obj instanceof TPEntity) {
			return ((TPEntity) obj).getName();
		}
		return obj.toString();
	}

	/**
	 * Gets the appropriate image corresponding to
	 * the given tree node object
	 *
	 * @param element the object
	 */
	public Image getImage(Object element) {
		Image image = null;
		TPEntity treeEnt = (TPEntity) element;
		/* Get the appropriate image based upon the
		 * the type of TPEntity
		 */
		if (treeEnt.getType().equals(TPEntity.FOLDER)) {
			image =
PlatformUI.getWorkbench().getSharedImages().getImage(
					ISharedImages.IMG_OBJ_FOLDER);
		} else if (treeEnt.getType().equals(TPEntity.JUNIT_TEST)) {
			image = getImageFromString("icons/junit.gif");
		} else if (treeEnt.getType().equals(TPEntity.EXEC_PASS)) {
			image = getImageFromString("icons/testok.gif");
		} else if (treeEnt.getType().equals(TPEntity.EXEC_FAIL)) {
			image = getImageFromString("icons/testfail.gif");
		} else if (treeEnt.getType().equals(TPEntity.EXEC_ERROR)) {
			image = getImageFromString("icons/testerr.gif");
		} else if
(treeEnt.getType().equals(TPEntity.EXEC_INCONCLUSIVE)) {
			image = getImageFromString("icons/testignored.gif");
		}
		return image;
	}

	/**
	 * Helper method to return an image based
	 * upon a given descriptor
	 * @param type the descriptor
	 * @return the image
	 */
	private Image getImageFromString(String type) {
		ImageDescriptor descriptor =
Activator.getImageDescriptor(type);
		// obtain the cached image corresponding to the descriptor
		Image image = (Image) imageCache.get(descriptor);
		if (image == null) {
			image = descriptor.createImage();
			imageCache.put(descriptor, image);
		}
		return image;
	}
}
```

TestView.java

```
/*****************************************************************
 *
 * File     :     TestView.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a view of the TPTeam Test Tree and hosts
 *                Actions that perform CRUD and executions
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import java.util.Hashtable;
import java.util.Observable;
import java.util.Observer;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.IToolBarManager;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.viewers.SelectionChangedEvent;
import org.eclipse.jface.viewers.TreeViewer;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.part.ViewPart;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.ITreeNode;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TreeNodeModel;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestExecutionXML;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs.AddFolderDialog;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs.AddTestDialog;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.dialogs.UpdateDialog;
import
edu.harvard.fas.rbrady.tpteam.tpbuddy.eventadmin.EventAdminHandler;

/*****************************************************************
 * File           :     TestView.java
 *
 * Description    :     Provides a view of the TPTeam Test Tree and
 *                      hosts Actions that perform CRUD and executions
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
```

```
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class TestView extends ViewPart implements Observer {
      /** The view ID */
      public static final String ID =
"edu.harvard.fas.rbrady.tpteam.tpbuddy.views.testview";
      /** An Action to request test executions */
      private Action mExecTest;
      /** An Action to request test deletions */
      private Action mDelTest;
      /** An Action to request test updates */
      private Action mUpdateTest;
      /** An Action to request test details */
      private Action mShowTest;
      /** An Action to request new test folders */
      private Action mAddFolder;
      /** An Action to request new test definitions */
      private Action mAddTest;
      /** The Test tree TreeViewer */
      private TreeViewer mViewer;
    /** The test tree content provider */
      private TestContentProvider mTestContentProvider;
      /** The model providing input to the content provider */
      private TreeNodeModel mTreeNodeModel;
      /** The TPTeam databasee ID of the project */
      private String mProjID;
      /** The TPTeam name of the project */
      private String mProjName;

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting execution of the selected node
       *
       * Performed when user clicks on exec test action icon in view
       */
      private void execTestAction() {
            // Get selected node in test tree
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            TPEntity treeEnt = (TPEntity) selection.getFirstElement();

            // Get parent shell to display dialog, if necessary
            Shell parent = getViewSite().getShell();

            // Throw error dialog if execution attempted on folder node
            if (treeEnt.getType().equals(TPEntity.FOLDER)) {
                  MessageDialog
                              .openError(parent, "Exec Test Error",
                                          "Execute Test Error: Folders
can not be executed as test definitions.");
                  return;
            }

            // Send TPEvent requesting test execution
```

```
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_EXEC_REQ_TOPIC, dictionary);
            sendMsgToEventAdmin(tpEvent);
      }

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting deletion of the selected node
       *
       * Performed when user clicks on delete test action icon in view
       */
      private void delTestAction() {

            // Get the selected node in test tree
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            final TPEntity treeEnt = (TPEntity)
selection.getFirstElement();

            // Get parent shell in case error dialog needs to be shown
            Shell parent = getViewSite().getShell();

            // Show error dialog if user attemped to delete test tree
root node
            if (Integer.parseInt(treeEnt.getID()) < 1) {
                  MessageDialog.openError(parent, "Delete Test Error",
                              "Delete Test Error: The root node can not
be deleted.");
                  return;
            }

            // Confirmation needed for deletions, for safety.
            boolean confirm = MessageDialog.openConfirm(parent,
                        "Delete Test Confirmation",
                        "Are you sure you want to delete test entity "
                                    + treeEnt.getName() + "?");
            if (!confirm)
                  return;

            // Wrap info into TPEvent and send delete request
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
```

```
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            TPEvent tpEvent = new TPEvent(ITPBridge.TEST_DEL_REQ_TOPIC,
dictionary);
            sendMsgToEventAdmin(tpEvent);
      }

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting update of the selected node.
       *
       * Launches an update dialog to collect data from user.
       *
       * Performed when user clicks on update test action icon in view
       */
      private void updateTestAction() {
            // Get the selected node in test tree
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            final TPEntity treeEnt = (TPEntity)
selection.getFirstElement();

            // Get parent shell in case error dialog needs to be shown
            Shell parent = getViewSite().getShell();

            // Show error dialog if user attempts to update test tree
root
            if (Integer.parseInt(treeEnt.getID()) < 1) {
                  MessageDialog.openError(parent, "Update Test Error",
                              "Update Test Error: The root node can not
be updated.");
                  return;
            }

            // Wrap update request into TPEvent and send
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_UPDATE_DATA_REQ_TOPIC,
                        dictionary);
            sendMsgToEventAdmin(tpEvent);

            UpdateDialog updateDialog = new UpdateDialog(parent,
treeEnt.getType()
                        .equals(TPEntity.FOLDER));

            if (updateDialog.open() == UpdateDialog.OK) {
```

```
                  String testXML =
TestXML.getXML(updateDialog.getTestStub());
                  System.out.println("testXML:\n" + testXML);
                  dictionary.put(TPEvent.TEST_XML_KEY, testXML);
                  tpEvent = new
TPEvent(ITPBridge.TEST_UPDATE_REQ_TOPIC, dictionary);
                  sendMsgToEventAdmin(tpEvent);
            }
      }

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting addition of a new folder using
       * the selected node as parent.
       *
       * Launches an add folder dialog to collect user input.
       *
       * Performed when user clicks on add test folder action icon in
view
       */
      private void addFolderAction() {
            // Get parent folder where new child folder will be added
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            final TPEntity treeEnt = (TPEntity)
selection.getFirstElement();

            // Get the parent shell so add folder dialog can be shown
            Shell parent = getViewSite().getShell();

            // Error checking
            if (!treeEnt.getType().equals(TPEntity.FOLDER)) {
                  MessageDialog
                              .openError(parent, "Add Test Error",
                                          "Add Test Error: Folders can
not be added to test definitions.");
                  return;
            }

            // Show add folder dialog and pull-out info
            AddFolderDialog addFolderDialog = new
AddFolderDialog(parent, Integer
                        .parseInt(treeEnt.getID()));

            if (addFolderDialog.open() == AddFolderDialog.OK) {
                  Test testStub = addFolderDialog.getTestStub();
                  TpteamUser addUser = new TpteamUser();

      addUser.setEcfId(Activator.getDefault().getTPBridgeClient()
                              .getTargetIDName());
                  testStub.setCreatedBy(addUser);
                  Project proj = new Project();
                  proj.setId(Integer.parseInt(mProjID));
                  testStub.setProject(proj);
                  String testXML = TestXML.getXML(testStub);
```

```
                  // Wrap info into TPEvent and send add folder request
                  Hashtable<String, String> dictionary = new
Hashtable<String, String>();
                  dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
                  dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
                  dictionary.put(TPEvent.SEND_TO,
Activator.getDefault()
                              .getTPBridgeClient().getTPMgrECFID());
                  dictionary.put(TPEvent.FROM, addUser.getEcfId());

                  TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_ADD_REQ_TOPIC,
                              dictionary);
                  dictionary.put(TPEvent.TEST_XML_KEY, testXML);
                  sendMsgToEventAdmin(tpEvent);
            }
      }

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting addition of a new test definition using
       * the selected node as parent.
       *
       * Launches an add test dialog to collect user input.
       *
       * Performed when user clicks on add test defiinition action icon
in view
       */
      private void addTestAction() {
            // Get parent folder where test definition will be added
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            final TPEntity treeEnt = (TPEntity)
selection.getFirstElement();

            // Get parent shell so add test dialog can be shown
            Shell parent = getViewSite().getShell();

            // Error check
            if (!treeEnt.getType().equals(TPEntity.FOLDER)) {
                  MessageDialog
                              .openError(parent, "Add Test Error",
                                          "Add Test Error: Test
definitions can not be added to test definitions.");
                  return;
            }

            // Show add dialog and pullout info
            AddTestDialog addTestDialog = new AddTestDialog(parent,
Integer
                        .parseInt(treeEnt.getID()));

            if (addTestDialog.open() == AddTestDialog.OK) {
                  Test testStub = addTestDialog.getTestStub();
                  TpteamUser addUser = new TpteamUser();
```

```
      addUser.setEcfId(Activator.getDefault().getTPBridgeClient()
                              .getTargetIDName());
                  testStub.setCreatedBy(addUser);
                  Project proj = new Project();
                  proj.setId(Integer.parseInt(mProjID));
                  testStub.setProject(proj);
                  String testXML = TestXML.getXML(testStub);

                  // Wrap info into TPEvent and send add test request
                  Hashtable<String, String> dictionary = new
Hashtable<String, String>();
                  dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
                  dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
                  dictionary.put(TPEvent.SEND_TO,
Activator.getDefault()
                              .getTPBridgeClient().getTPMgrECFID());
                  dictionary.put(TPEvent.FROM, addUser.getEcfId());

                  TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_ADD_REQ_TOPIC,
                              dictionary);
                  dictionary.put(TPEvent.TEST_XML_KEY, testXML);
                  sendMsgToEventAdmin(tpEvent);
            }
      }

      /**
       * Extracts node information from tree, wraps into
       * TPEvent requesting details of the selected test
       * node.
       *
       * Performed when user clicks on test details action icon in view
       */
      private void showTestAction() {
            // Get selected node
            IStructuredSelection selection = (IStructuredSelection)
mViewer
                        .getSelection();
            final TPEntity treeEnt = (TPEntity)
selection.getFirstElement();

            // Wrap info into TPEvent and send test detail request
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY,
String.valueOf(treeEnt.getID()));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_DETAIL_REQ_TOPIC,
                        dictionary);
            sendMsgToEventAdmin(tpEvent);
```

```
            // Give focus to test detail view now
            showDetailView();
      }

      /**
       * Gives focus to test detail view while waiting for test detail
results to
       * arrive.
       */
      private void showDetailView() {
            IWorkbenchPage page = PlatformUI.getWorkbench()
                        .getActiveWorkbenchWindow().getActivePage();
            if (page != null) {
                  try {
                        page.showView(DetailView.ID);
                  } catch (PartInitException e) {

                        e.printStackTrace();
                  }
            }
      }

      /**
       * Submits a TPEvent that request a response event containing
       * the entire project test tree in serialized XML form.
       *
       */
      private void getTestTreeAction() {
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTargetIDName());
            dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
            dictionary.put(TPEvent.PROJECT_KEY, mProjName);

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_TREE_GET_REQ_TOPIC, dictionary);
      }

      /**
       *  Sends a TPEvent to the EventAdmin, which will submit
       *  to the OSGi Event service and route to all interested
       *  subscribers
       *
       * @param tpEvent
       */
      private void sendMsgToEventAdmin(TPEvent tpEvent) {
            Activator.getDefault().getEventAdminClient().sendEvent(
                        tpEvent.getTopic(), tpEvent.getDictionary());
      }

      /**
       * Helper method to enable the various test tree
       * actions a user can carry out through the view
```

```
	 */
	private void createActions() {
		mExecTest.setEnabled(true);
		mExecTest.setImageDescriptor(Activator
				.getImageDescriptor("icons/runjunit.gif"));

		mDelTest.setEnabled(true);
		mDelTest.setImageDescriptor(Activator
				.getImageDescriptor("icons/delete.gif"));

		mShowTest.setEnabled(true);
		mShowTest.setImageDescriptor(Activator
				.getImageDescriptor("icons/test.gif"));

		mUpdateTest.setEnabled(true);
		mUpdateTest.setImageDescriptor(Activator
				.getImageDescriptor("icons/update_tree.gif"));

		mAddFolder.setEnabled(true);
		mAddFolder.setImageDescriptor(Activator

	.getImageDescriptor("icons/folderadd_pending.gif"));

		mAddTest.setEnabled(true);
		mAddTest.setImageDescriptor(Activator
				.getImageDescriptor("icons/new_testcase.gif"));

		mViewer.addSelectionChangedListener(new
ISelectionChangedListener() {
			public void selectionChanged(SelectionChangedEvent
event) {
				// updateAction();
			}
		});

		IToolBarManager mgr =
getViewSite().getActionBars().getToolBarManager();
		mgr.add(mShowTest);
		mgr.add(mAddFolder);
		mgr.add(mAddTest);
		mgr.add(mUpdateTest);
		mgr.add(mDelTest);
		mgr.add(mExecTest);
	}

	/**
	 * This is a callback that will allow us to create the mViewer
and
	 * initialize it.
	 */
	public void createPartControl(Composite parent) {


	Activator.getDefault().getEventAdminHandler().addObserver(this);

		mTreeNodeModel = new TreeNodeModel();
```

```
mExecTest = new Action("Run...") {
	public void run() {
		execTestAction();
	}
};

mDelTest = new Action("Delete...") {
	public void run() {
		delTestAction();
	}
};

mShowTest = new Action("Details...") {
	public void run() {
		showTestAction();
	}
};

mUpdateTest = new Action("Update...") {
	public void run() {
		updateTestAction();
	}
};

mAddFolder = new Action("Add Test Folder...") {
	public void run() {
		addFolderAction();
	}
};

mAddTest = new Action("Add Test Definition...") {
	public void run() {
		addTestAction();
	}
};

mViewer = new TreeViewer(parent, SWT.MULTI | SWT.H_SCROLL
		| SWT.V_SCROLL | SWT.BORDER);
mTestContentProvider = new TestContentProvider();
mViewer.setContentProvider(mTestContentProvider);
mViewer.setLabelProvider(new TestLabelProvider());
try {
	mViewer.setInput(null/* TestXML.getExample() */);
} catch (Exception e) {
	e.printStackTrace();
}
getSite().setSelectionProvider(mViewer);

createActions();

mProjID = Activator.getDefault().getProjID();
mProjName = Activator.getDefault().getProjName();

if (mProjID != null && !mProjID.equalsIgnoreCase("")
		&& Integer.parseInt(mProjID) > 0)
	getTestTreeAction();
```

```
	}

	/**
	 * Passing the focus request to the mViewer's control.
	 */
	public void setFocus() {
		mViewer.getControl().setFocus();
	}

	/**
	 * Update used to refresh view when a test
	 * response TPEvent is received
	 *
	 * @param observable the object issuing the update
	 * @param object the TPEvent to be handled
	 */
	public void update(Observable observable, Object object) {
		if (observable instanceof EventAdminHandler
				&& object instanceof TPEvent) {
			TPEvent tpEvent = (TPEvent) object;

			if
(tpEvent.getTopic().equals(ITPBridge.TEST_EXEC_RESULT_TOPIC)) {
				updateExecution(tpEvent);
			}
			else if (tpEvent.getTopic().equalsIgnoreCase(
					ITPBridge.TEST_TREE_GET_REQ_TOPIC)) {
				mProjID =
tpEvent.getDictionary().get(TPEvent.PROJECT_ID_KEY);
				mProjName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);
				Activator.getDefault().setProjID(mProjID);
				Activator.getDefault().setProjName(mProjName);
			}
			else if (((TPEvent) object).getTopic().equals(
					ITPBridge.CHART_GET_DATA_REQ_TOPIC)) {
				mProjID =
tpEvent.getDictionary().get(TPEvent.PROJECT_ID_KEY);
				mProjName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);
				Activator.getDefault().setProjID(mProjID);
				Activator.getDefault().setProjName(mProjName);
			}
			else if (tpEvent.getTopic().equalsIgnoreCase(
					ITPBridge.TEST_TREE_GET_RESP_TOPIC)) {
				String testTreeXML =
tpEvent.getDictionary().get(
						TPEvent.TEST_TREE_XML_KEY);
				System.out.println(testTreeXML);
				final TPEntity projRoot = TestXML
					.getTPEntityFromXML(testTreeXML);

				mTreeNodeModel.clear();
				populateModel(projRoot);

				Display.getDefault().syncExec(new Runnable() {
					public void run() {
```

```
                                    mViewer
                                                .setInput(new
TPEntity[] { projRoot } /* projRoot.getChildren() */);
                              }
                        });
                  }
                  else if (tpEvent.getTopic().equalsIgnoreCase(
                              ITPBridge.TEST_DEL_RESP_TOPIC)) {
                        final String nodeID = tpEvent.getID();
                        Display.getDefault().syncExec(new Runnable() {
                              public void run() {
                                    ITreeNode delNode =
mTreeNodeModel.get(nodeID);
                                    ITreeNode parent;
                                    if (delNode != null
                                                && (parent =
delNode.getParent()) != null) {
                                          parent.removeChild(delNode);
                                    }
                              }
                        });
                  }
                  else if (tpEvent.getTopic().equalsIgnoreCase(
                              ITPBridge.TEST_UPDATE_RESP_TOPIC)) {
                        final String nodeID = tpEvent.getID();
                        final String testName =
tpEvent.getDictionary().get(
                                    TPEvent.TEST_NAME_KEY);
                        Display.getDefault().syncExec(new Runnable() {
                              public void run() {
                                    ITreeNode updateNode =
mTreeNodeModel.get(nodeID);
                                    updateNode.setName(testName);
                              }
                        });
                  }
                  else if (tpEvent.getTopic().equalsIgnoreCase(
                              ITPBridge.TEST_ADD_RESP_TOPIC)) {
                        String testXML = tpEvent.getDictionary().get(
                                    TPEvent.TEST_XML_KEY);
                        System.out.println("testAddXML:\n" + testXML);
                        final Test testStub =
TestXML.getTestFromXML(testXML);
                        final TPEntity tpEntity =
TestXML.getTPEntityFromTest(testStub);
                        Display.getDefault().syncExec(new Runnable() {
                              public void run() {
                                    ITreeNode addNode = tpEntity;
                                    ITreeNode parent = null;
                                    if (testStub.getParent().getId() ==
0) {
                                          parent =
mTreeNodeModel.get("0");
                                    } else {
                                          parent =
mTreeNodeModel.get(String.valueOf(testStub
```

```
      .getParent().getId()));
                                    }
                                    if (parent != null) {
                                          addNode.setParent(parent);
                                          parent.addChild(addNode);

      mTreeNodeModel.addNode(addNode);
                                    }
                              }
                        });

                  }
            }
      }

      /**
       * Populates the view's model with a root
       * ITreeNode
       * @param treeNode the root ITreeNode
       */
      private void populateModel(ITreeNode treeNode) {
            mTreeNodeModel.put(treeNode.getID(), treeNode);
            for (ITreeNode child : treeNode.getChildren())
                  populateModel(child);
      }

      /**
       * Helper method used to add a test execution
       * as a child to a test definition tree node
       *
       * @param tpEvent the TPEvent containing the execution
       */
      private void updateExecution(final TPEvent tpEvent) {
            Display.getDefault().syncExec(new Runnable() {
                  public void run() {
                        String execXML = tpEvent.getDictionary().get(
                                    TPEvent.TEST_EXEC_XML_KEY);
                        ITreeNode execNode = TestExecutionXML
                                    .getTPEntityFromXML(execXML);
                        ITreeNode parent = null;
                        if ((parent =
mTreeNodeModel.get(execNode.getParent().getID())) != null) {
                              execNode.setParent(parent);
                              parent.addChild(execNode);
                        }
                  }
            });
      }

      public void dispose() {
            super.dispose();

      Activator.getDefault().getEventAdminHandler().deleteObserver(this
);
      }
}
```

TPEventLabelProvider.java

```
/*****************************************************************
 *
 * File     :     TPEventLabelProvider.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides the text and images for the columns in the
 *                EventHistoryView table
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.views;

import org.eclipse.jface.viewers.ITableLabelProvider;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.swt.graphics.Image;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*****************************************************************
 * File           :     TPEventLabelProvider.java
 *
 * Description    :     Provides the text and images for the columns in
 *                      the EventHistoryView table
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TPEventLabelProvider extends LabelProvider implements
            ITableLabelProvider {

      /**
       * Extracts table column text
       * @param element the Object to extract text from
       * @param index the column index
       */
      public String getColumnText(Object element, int index) {
            TPEvent tpEvent = (TPEvent) element;
            switch (index) {
            case 0:
                  return tpEvent.getDictionary().get(TPEvent.FROM);
            case 1:
                  return tpEvent.getDictionary().get(TPEvent.SEND_TO);
            case 2:
                  String[] topicPath = tpEvent.getTopic().split("/");
                  return topicPath[topicPath.length-1];
            case 3:
                  return tpEvent.getProject();
            case 4:
                  return tpEvent.getTestName();
            case 5:
                  return tpEvent.getID();
            case 6:
                  return tpEvent.getStatus();
            default:
                  return "unknown " + index;
```

```
            }
        }

        /**
         * Gets the appropriate column image
         *
         * @param element the object to get column image
         * @param columnIndex the column index
         */
        public Image getColumnImage(Object element, int columnIndex) {
            return null;
        }
}
```

package edu.harvard.fas.rbrady.tpteam.tpbuddy.wizard

XMPPConnectWizard.java

```
/*****************************************************************
 *
 * File     :     XMPPConnectWizard.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a Wizard dialog for connecting to an XMPP
 *                account
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.wizard;

import java.util.StringTokenizer;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.identity.IDCreateException;
import org.eclipse.ecf.core.identity.IDFactory;
import org.eclipse.ecf.core.security.ConnectContextFactory;
import org.eclipse.ecf.core.security.IConnectContext;
import org.eclipse.ecf.core.user.User;
import org.eclipse.ecf.presence.IPresenceContainerAdapter;
import org.eclipse.ecf.presence.ui.PresenceUI;
import org.eclipse.ecf.ui.IConnectWizard;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.jface.wizard.Wizard;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.IWorkbench;
import org.jivesoftware.smack.GoogleTalkConnection;
import org.jivesoftware.smack.XMPPConnection;
import org.jivesoftware.smack.XMPPException;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.Activator;
import edu.harvard.fas.rbrady.tpteam.tpbuddy.tpbridge.TPBridgeClient;

/*****************************************************************
 * File           :     XMPPConnectWizard.java
 *
 * Description    :     Provides a Wizard dialog for connecting to an
 *                      XMPP account
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class XMPPConnectWizard extends Wizard implements IConnectWizard
{

      /** Default ECF Connection Container Type */
      public static final String CONTAINER_TYPE = "ecf.xmpps.smack";
      /** Default Google Talk Server */
      public static final String GOOGLE_TALK_HOST = "gmail.com";
```

```
      /** The XMPPConnectWizard page used for collecting data */
      XMPPConnectWizardPage page;
    /** The parent Shell */
      private Shell shell;
      /** The ECF Communication Container to be connected */
      private IContainer container;
      /** Ecf ID of user */
      private ID targetID;
      /** User ECF Security Context */
      private IConnectContext connectContext;

      /**
       * Adds Wizard pages to the Wizard
       */
      public void addPages() {
            page = new XMPPConnectWizardPage();
            addPage(page);
      }

      /**
       * Initialize the Wizard
       */
      public void init(IWorkbench workbench, IContainer container) {
            shell = workbench.getActiveWorkbenchWindow().getShell();
            this.container = container;
      }

      /**
       * Extract userName@hostName and password from
       * Wizard page, then connect and display buddy roster
       */
      public boolean performFinish() {
            connectContext = ConnectContextFactory

      .createPasswordConnectContext(page.getPassword());

            try {
                  targetID = IDFactory.getDefault().createID(
                              container.getConnectNamespace(),
page.getConnectID());
            } catch (final IDCreateException e) {
                  Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                              MessageDialog.openError(shell, "TPBuddy
Login Error",
                                          "ConnectionException: " +
e.getMessage());
                        }
                  });
                  e.printStackTrace();
                  return false;
            }
            // Get presence container adapter
            IPresenceContainerAdapter presenceAdapter =
(IPresenceContainerAdapter) container
                        .getAdapter(IPresenceContainerAdapter.class);
            // Create and show roster view user interface
```

```
            new PresenceUI(container, presenceAdapter).showForUser(new
User(
                        targetID));

            // Extract userName & hostName for quick connection
validation
            StringTokenizer st = new
StringTokenizer(targetID.getName(), "@");
            String userName = st.nextToken();
            String hostName = st.nextToken();
            boolean isConnValid = false;
            if (hostName.equalsIgnoreCase(GOOGLE_TALK_HOST)) {
                  isConnValid = isGoogleTalkConnValid(userName,
page.getPassword());
            } else {
                  isConnValid = isXMPPConnValid(hostName, userName,
page
                              .getPassword());
            }

            // If connection not valid, exit to avoid hanging SSL
            if (!isConnValid)
                  return true;

            // Proceed to bridge
            final TPBridgeClient bridgeClient = Activator.getDefault()
                        .getTPBridgeClient();
            final IContainer bridgeContainer = this.container;
            final ID bridgeTargetID = this.targetID;
            final IConnectContext bridgeContext = this.connectContext;
            Display.getCurrent().asyncExec(new Runnable() {
                  public void run() {
                        bridgeClient.connect(bridgeContainer,
bridgeTargetID,
                                    bridgeContext);
                  }
            });

            return true;
      }

      /**
       * Helper method to determine if a Google Talk
       * connection userName/Password is a valid pair
       *
       * @param userName the name of the Google Talk account
       * @param password the password of the Google Talk account
       * @return true if the connection is valid, false otherwise
       */
      private boolean isGoogleTalkConnValid(String userName, String
password) {
            GoogleTalkConnection gtc;
            try {
                  gtc = new GoogleTalkConnection();
                  gtc.login(userName, page.getPassword());
                  gtc.close();
            } catch (final XMPPException e) {
```

```
                  e.printStackTrace();
                  MessageDialog.openError(shell, "TPBuddy Login Error",
                              "ConnectionException: " +
e.getMessage());
                  return false;
            }
            return true;
      }

      /**
       * Helper method to determine if a generic XMPP
       * connection userName/Password is a valid pair
       * for a given hostName
       *
       * @param hostName name of the XMPP server host
       * @param userName name of the XMPP account
       * @param password password of the XMPP account
       * @return true if the connection is valid, false otherwise
       */
      private boolean isXMPPConnValid(String hostName, String userName,
                  String password) {
            XMPPConnection XMPP;
            try {
                  XMPP = new XMPPConnection(hostName);
                  XMPP.login(userName, page.getPassword());
                  XMPP.close();
            } catch (final XMPPException e) {
                  e.printStackTrace();
                  MessageDialog.openError(shell, "TPBuddy Login Error",
                              "ConnectionException: " +
e.getMessage());
                  return false;
            }
            return true;
      }

}
```

XMPPConnectWizardPage.java

```
/*********************************************************************
 *
 * File     :     XMPPConnectWizardPage.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a Wizard page for connecting to an XMPP
 *                account
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpbuddy.wizard;

import org.eclipse.jface.dialogs.IDialogSettings;
import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.ModifyEvent;
import org.eclipse.swt.events.ModifyListener;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;

/*********************************************************************
* File            :     XMPPConnectWizardPage.java
*
* Description     :     Provides a Wizard page for connecting to an
*                       XMPP account
*
* @author Bob Brady, rpbrady@gmail.com
* @version $Revision$
* @date $Date$ Copyright (c) 2007 Bob Brady
*********************************************************************/
public class XMPPConnectWizardPage extends WizardPage {

      /** XMPP account connection String, e.g., user@server.com */
      Text connectText;
      /** XMPP account password */
      Text passwordText;
      /** Classname of companion Wizard */
      protected static final String CLASSNAME =
XMPPConnectWizardPage.class
                  .getName();
      /** Settings key for saving/retrieving dialog setting */
      private static final String DIALOG_SETTINGS = CLASSNAME;

      /**
       * Constructor
       */
      XMPPConnectWizardPage() {
            super("");
            setTitle("XMPPS Connection Wizard");
            setDescription("Specify a XMPP account to connect to.");
            setPageComplete(false);
      }
```

```
	/**
	 * Creates the displayed Composite for the page
	 */
	public void createControl(Composite parent) {
		parent.setLayout(new GridLayout());
		GridData fillData = new GridData(SWT.FILL, SWT.CENTER,
true, false);
		GridData endData = new GridData(SWT.FILL, SWT.CENTER, true,
false, 2, 1);

		Label label = new Label(parent, SWT.LEFT);
		label.setText("User ID:");

		connectText = new Text(parent, SWT.SINGLE | SWT.BORDER);
		restoreDialogSettings();
		connectText.setLayoutData(fillData);
		connectText.addModifyListener(new ModifyListener() {
			public void modifyText(ModifyEvent e) {
				if (!connectText.getText().equals("")) {
					updateStatus(null);
				} else {
					updateStatus("An connect ID must be
specified.");
				}
			}
		});

		label = new Label(parent, SWT.RIGHT);
		label.setText("<user>@<xmppserver>[:port]");
		label.setLayoutData(endData);

		label = new Label(parent, SWT.LEFT);
		label.setText("Password:");
		passwordText = new Text(parent, SWT.SINGLE | SWT.PASSWORD |
SWT.BORDER);
		passwordText.setLayoutData(fillData);

		setControl(parent);
	}


	protected void updateStatus(String message) {
		setErrorMessage(message);
		setPageComplete(message == null);
	}

	private void restoreDialogSettings() {
		IDialogSettings dialogSettings = getDialogSettings();
		if (dialogSettings != null) {
			IDialogSettings pageSettings = dialogSettings
					.getSection(DIALOG_SETTINGS);
			if (pageSettings != null) {
				String strVal =
pageSettings.get("connectText");
				if (strVal != null) {
					connectText.setText(strVal);
```

```
                        }
                  }
            }
      }

      public void saveDialogSettings() {
            IDialogSettings dialogSettings = getDialogSettings();
            if (dialogSettings != null) {
                  IDialogSettings pageSettings = dialogSettings
                              .getSection(DIALOG_SETTINGS);
                  if (pageSettings == null)
                        pageSettings =
dialogSettings.addNewSection(DIALOG_SETTINGS);

                  pageSettings.put("connectText",
connectText.getText());
            }
      }

      // Accessors

      String getConnectID() {
            return connectText.getText();
      }

      String getPassword() {
            return passwordText.getText();
      }


}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager

Activator.java

```
/*********************************************************************
 *
 * File     :     Activator.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Controls the lifecycle of the TPManager Plug-in
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager;

import org.hibernate.SessionFactory;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.util.tracker.ServiceTracker;

import
edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin.EventAdminClient;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin.EventAdminHandler;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.HttpServiceTracker;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tpbridge.TPBridgeClient;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tptp.TPManager;

/*********************************************************************
 * File           :     Activator.java
 *
 * Description    :     Controls the lifecycle of the TPManager Plug-in
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$
 * Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class Activator implements BundleActivator {

      /** The plug-in shared instance */
      private static Activator mBundle;

      private EventAdminClient mEventAdminClient;

      private EventAdminHandler mEventAdminHandler;

      private TPBridgeClient mTPBridgeClient;

      private TPManager mTPManager;

      private ServiceTracker mHttpServiceTracker;

      /** The database session */
      private SessionFactory mHiberSessionFactory;
```

```
	public Activator() {
		mBundle = this;
	}

	/**
	 * Called when plug-in starts its lifecycle
	 *
	 * Initializes the EventAdmin Service handler and client,
	 * sets the Hibernate database session
	 *
	 * @param context the plug-in context
	 */
	public void start(BundleContext context) throws Exception {
		mEventAdminHandler = new EventAdminHandler(context);
		mEventAdminClient = new EventAdminClient(context);
		mTPBridgeClient = new TPBridgeClient(context);
		mTPManager = new TPManager();
		mEventAdminHandler.addObserver(mTPManager);

		mHttpServiceTracker = new HttpServiceTracker(context);
		mHttpServiceTracker.open();

		mHiberSessionFactory =
mTPBridgeClient.getHibernateSessionFactory();
	}

	/*
	 * Closes the OSGi ServiceTracker and Hibernate
	 * database session
	 *
	 * @param context the plug-in context
	 */
	public void stop(BundleContext context) throws Exception {
		mHttpServiceTracker.close();
		mHttpServiceTracker = null;
		mHiberSessionFactory.close();
	}

	/**
	 * Returns the shared instance
	 *
	 * @return the shared instance
	 */
	public static Activator getDefault() {
		return mBundle;
	}

	// Public accessors

	public EventAdminClient getEventAdminClient() {
		return mEventAdminClient;
	}

	public EventAdminHandler getEventAdminHandler() {
		return mEventAdminHandler;
	}
```

```
	public TPBridgeClient getTPBridgeClient() {
		return mTPBridgeClient;
	}

	public TPManager getTPManager() {
		return mTPManager;
	}

	public SessionFactory getHiberSessionFactory() {
		return mHiberSessionFactory;
	}

}
```

. package edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin

EventAdminClient.java

```
/**********************************************************************
 *
 * File     :     EventAdminClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A client of the EventAdmin Service
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin;

import java.util.Hashtable;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventAdmin;
import org.osgi.util.tracker.ServiceTracker;

/**********************************************************************
 * File           :     EventAdminClient.java
 *
 * Description    :     A client of the EventAdmin Service.  It sends
 *                      TPTeam Event data encapsulated as an OSGi Event
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class EventAdminClient {

      /** The EventAdmin ServiceTracker */
      private ServiceTracker mServiceTracker;

      /**
       * Constructor, gets and opens a tracker for
       * the EventAdmin OSGi Service
       *
       * @param context The TPManager plug-in context
       */
      public EventAdminClient(BundleContext context) {
            mServiceTracker = new ServiceTracker(context,
EventAdmin.class
                        .getName(), null);
            mServiceTracker.open();
      }

      /**
       * Sends an OSGi Event to the OSGi EventAdmin Service
       * @param topic the event topic String
       * @param dictionary a table of key=value pairs
       * @return true if message sent successfully, false otherwise
       */
```

```
      public boolean sendEvent(String topic, Hashtable<String, String>
dictionary) {
            boolean messageSent = false;

            EventAdmin eventAdmin = (EventAdmin)
mServiceTracker.getService();

            if (eventAdmin != null) {
                  System.out.println("EventAdminClient: Sent " +
topic);
                  eventAdmin.sendEvent(new Event(topic, dictionary));
                  messageSent = true;
            }
            return messageSent;
      }
}
```

EventAdminHandler.java

```
/**********************************************************************
 *
 * File     :     EventAdminHandler.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The OSGi Service that handles OSGi Events from the
 *                EventAdmin Service and acts as an observable of
 *                TPTeam Events.  Notifies the TPManager object when
 *                events are received.
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Observable;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import org.osgi.framework.BundleContext;
import org.osgi.service.event.Event;
import org.osgi.service.event.EventConstants;
import org.osgi.service.event.EventHandler;

/**********************************************************************
 * File           :     EventAdminHandler.java
 *
 * Description    :     The OSGi Service that handles OSGi Events from
 *                      the EventAdmin Service and acts as an
 *                      observable of TPTeam Events.  Notifies the
 *                      TPManager object when events are received.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class EventAdminHandler extends Observable implements
EventHandler {
      /** Table of key=value pairs of the OSGi Event */
      private Hashtable<String, String[]> mDictionary = new
Hashtable<String, String[]>();
      /** A List log of all TPEvents received */
      private ArrayList<Event> mEvents;

      /**
       * Constructor
       *
       * Sets the OSGi Event topics to be handled by this service
       *
       * Sets the
       * @param context
       */
      public EventAdminHandler(BundleContext context) {
            mEvents = new ArrayList<Event>();
            mDictionary.put(EventConstants.EVENT_TOPIC, new String[] {
```

```
                        ITPBridge.TEST_EXEC_REQ_TOPIC,
ITPBridge.PROJ_GET_REQ_TOPIC,
                        ITPBridge.TEST_TREE_GET_REQ_TOPIC,
ITPBridge.TEST_DEL_REQ_TOPIC,
                        ITPBridge.TEST_DETAIL_REQ_TOPIC,
ITPBridge.TEST_UPDATE_DATA_REQ_TOPIC,
                        ITPBridge.TEST_UPDATE_REQ_TOPIC,
ITPBridge.TEST_ADD_REQ_TOPIC,
                        ITPBridge.CHART_GET_DATA_REQ_TOPIC});
            context
                        .registerService(EventHandler.class.getName(),
this,
                                    mDictionary);
            System.out
                        .println("TPManager EventAdminHandler:
registered EventAdmin service");
      }

      /**
       * Handles an OSGi Event from the EventAdmin Service
       * and notifies the TPManager so that it can
       * perform the necessary CRUD or execution operation.
       *
       * @param event the OSGi Event to be handled
       */
      public void handleEvent(Event event) {
            System.out.println("TPManager EventAdminHandler: Got "
                        + event.getTopic() + " Event");
            TPEvent tpEvent = new TPEvent(event);
            setChanged();
            notifyObservers(tpEvent);
      }

      /**
       * Getter
       * @return the List log of TPTeam Events
       */
      public ArrayList<Event> getEventLog() {
            return mEvents;
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate

ChartUtil.java

```
/*********************************************************************
 *
 * File     :     ChartUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A TPTeam project chart utility class.
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataPoint;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.ChartDataSetXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;

/*********************************************************************
 * File           :     ChartUtil.java
 *
 * Description    :     A TPTeam project chart utility class.  Contains
 *                      methods for creating data points from Hibernate
 *                      database calls.  Serializes charts into XML
 *                      Strings.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ChartUtil {

      /** Number of history days to display on line charts */
      public static final int NUM_LINECHART_DAYS = 30;

      /**
       * Gets a data point for a single pie chart.  The data point
       * contains the number of test execution successes, failures,
       * errors, inconclusive, and not-executed results for a given
       * project.
       *
```

```
       * @param tpEvent the TPEvent containing project metadata
       * @return the ChartDataPoint
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private static ChartDataPoint getPieChartDataPoint(TPEvent
tpEvent)
                  throws Exception {
            Session s = null;
            Transaction tx = null;
            int projID = Integer.parseInt(tpEvent.getDictionary().get(
                        TPEvent.PROJECT_ID_KEY));
            ChartDataPoint dataPoint = new ChartDataPoint();
            try {

                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();
                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }

                  tx = s.beginTransaction();

                  String SQL_QUERY = "SELECT te.status, count(*) FROM
TestExecution as te "
                              + " join te.test as t "
                              + " WHERE t.project.id = ? AND t.isFolder
= 'N' "
                              + " AND (te.execDate = "
                              + " (SELECT max(te2.execDate) from
TestExecution te2 where te2.test.id = te.test.id) "
                              + " or te.execDate is NULL) GROUP BY
te.status";

                  Query query = s.createQuery(SQL_QUERY);
                  query.setInteger(0, projID);

                  int sum = 0;
                  for (Iterator it = query.iterate(); it.hasNext();) {
                        Object[] row = (Object[]) it.next();
                        Character status = (Character) row[0];
                        Long count = (Long) row[1];

                        if (status == 'P')
                              dataPoint.setPass(count.intValue());
                        else if (status == 'F')
                              dataPoint.setFail(count.intValue());
                        else if (status == 'E')
                              dataPoint.setError(count.intValue());
                        else if (status == 'I')
                              dataPoint.setInconcl(count.intValue());
                        sum += count.intValue();
                  }
```

```
                  SQL_QUERY = "SELECT count(*) FROM Test as t "
                              + " WHERE t.project.id = ? AND t.isFolder
= 'N' ";
                  query = s.createQuery(SQL_QUERY);
                  query.setInteger(0, projID);

                  for (Iterator it = query.iterate(); it.hasNext();) {
                        Long totalTests = (Long) it.next();
                        int notExec = totalTests.intValue() - sum;
                        dataPoint.setNotExec(notExec);
                  }

                  tx.commit();
            } catch (Exception e) {
                  e.printStackTrace();
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }

            return dataPoint;
      }

      /**
       * Gets the String XML serialization for a pie chart
       *
       * @param tpEvent the TPEvent holding the project metadata
       * @return the XML serialization String
       * @throws Exception
       */
      public static String getPieChartXML(TPEvent tpEvent) throws
Exception {
            ChartDataPoint dataPoint = getPieChartDataPoint(tpEvent);
            ChartDataSet dataSet = new ChartDataSet();
            TpteamUser user = new TpteamUser();
            user.setFirstName("root");
            user.setLastName("root");
            dataSet.setUser(user);

      dataSet.setProjName(tpEvent.getDictionary().get(TPEvent.PROJECT_K
EY));
            dataSet.addChartDataPoint(dataPoint);
            dataSet.setType(ChartDataSet.PIE);
            return ChartDataSetXML.getXML(dataSet);
      }

      /**
       * Gets the String XML serialization for a bar chart
       *
       * @param tpEvent the TPEvent holding the project metadata
       * @return the XML serialization String
       * @throws Exception
       */
      public static String getBarChartXML(TPEvent tpEvent) throws
Exception {
            List<ChartDataSet> dataSetList = new
ArrayList<ChartDataSet>();
```

```
            int projID = Integer.parseInt(tpEvent.getDictionary().get(
                        TPEvent.PROJECT_ID_KEY));
            String projName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);
            TpteamUser[] projUsers = getProjUsers(projID);

            for (TpteamUser projUser : projUsers) {
                  ChartDataSet dataSet = new ChartDataSet();
                  dataSet.setUser(projUser);
                  dataSet.setProjName(projName);
                  dataSet.setType(ChartDataSet.BAR);
                  ChartDataPoint dataPoint =
getBarChartDataPoint(projID, projUser
                              .getId());
                  dataSet.addChartDataPoint(dataPoint);
                  dataSetList.add(dataSet);
            }

            return ChartDataSetXML.getListXML(dataSetList);
      }

      /**
       * Gets a bar chart data point for a particular user.
       * The data point contains the number of test execution
       * successes, failures, errors, inconclusive, and
       * not-executed results associated with the TPTeam user
       * of a given project.
       *
       * @param projID the TPTeam ID of the project
       * @param userID the TPteam ID of the user
       * @return the ChartDataPoint
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private static ChartDataPoint getBarChartDataPoint(int projID,
int userID)
                  throws Exception {
            Session s = null;
            Transaction tx = null;
            ChartDataPoint dataPoint = new ChartDataPoint();
            try {

                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();
                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }

                  tx = s.beginTransaction();

                  String SQL_QUERY = "SELECT te.status, count(*) FROM
TestExecution as te "
                              + " join te.test as t "
```

```
                              + " WHERE t.project.id = ? AND
t.createdBy = ? AND t.isFolder = 'N' "
                              + " AND (te.execDate = "
                              + " (SELECT max(te2.execDate) from
TestExecution te2 where te2.test.id = te.test.id) "
                              + " or te.execDate is NULL) GROUP BY
te.status";

                  Query query = s.createQuery(SQL_QUERY);
                  query.setInteger(0, projID);
                  query.setInteger(1, userID);

                  int sum = 0;
                  for (Iterator it = query.iterate(); it.hasNext();) {
                        Object[] row = (Object[]) it.next();
                        Character status = (Character) row[0];
                        Long count = (Long) row[1];

                        if (status == 'P')
                              dataPoint.setPass(count.intValue());
                        else if (status == 'F')
                              dataPoint.setFail(count.intValue());
                        else if (status == 'E')
                              dataPoint.setError(count.intValue());
                        else if (status == 'I')
                              dataPoint.setInconcl(count.intValue());
                        sum += count.intValue();
                  }

                  SQL_QUERY = "SELECT count(*) FROM Test as t "
                              + " WHERE t.project.id = ? AND
t.createdBy = ? AND t.isFolder = 'N' ";
                  query = s.createQuery(SQL_QUERY);
                  query.setInteger(0, projID);
                  query.setInteger(1, userID);

                  for (Iterator it = query.iterate(); it.hasNext();) {
                        Long totalTests = (Long) it.next();
                        int notExec = totalTests.intValue() - sum;
                        dataPoint.setNotExec(notExec);
                  }
                  tx.commit();
            } catch (Exception e) {
                  e.printStackTrace();
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return dataPoint;
      }

      /**
       * Helper method to return the TPTeam users associated
       * with the given project.
       *
       * @param projID the ID of the project
       * @return an array of the associated TPTeam users
```

```
	 * @throws Exception
	 */
	public static TpteamUser[] getProjUsers(int projID) throws
Exception {
		List<TpteamUser> projUsers = new ArrayList<TpteamUser>();
		Session s = null;
		// Use plugin activator if in OSGi runtime
		if (Activator.getDefault() != null) {
			s = Activator.getDefault().getHiberSessionFactory()
					.getCurrentSession();
		} else {
			s =
HibernateUtil.getSessionFactory().getCurrentSession();
		}
		Transaction tx = null;
		try {
			tx = s.beginTransaction();
			Project proj = (Project) s.load(Project.class,
projID);
			for (TpteamUser user : proj.getTpteamUsers()) {
				TpteamUser projUser = new TpteamUser();
				projUser.setId(user.getId());
				projUser.setLastName(user.getLastName());
				projUser.setFirstName(user.getFirstName());
				projUser.setEcfId(user.getEcfId());
				projUsers.add(projUser);
			}
			tx.commit();
		} catch (Exception e) {
			e.printStackTrace();
			if (tx != null)
				tx.rollback();
			throw e;
		}
		return (TpteamUser[]) projUsers
				.toArray(new TpteamUser[projUsers.size()]);
	}

	/**
	 * Gets the String XML serialization for a line chart
	 *
	 * @param tpEvent the TPEvent holding the project metadata
	 * @return the XML serialization String
	 * @throws Exception
	 */
	public static String getLineChartXML(TPEvent tpEvent) throws
Exception {
		int projID = Integer.parseInt(tpEvent.getDictionary().get(
				TPEvent.PROJECT_ID_KEY));
		String projName =
tpEvent.getDictionary().get(TPEvent.PROJECT_KEY);

		ChartDataSet dataSet = new ChartDataSet();
		dataSet.setProjName(projName);
		dataSet.setType(ChartDataSet.LINE);

		Calendar cal = Calendar.getInstance();
```

```
            Date today = new Date();
            for (int idx = 0; idx < NUM_LINECHART_DAYS; idx++) {
                  cal.setTime(today);
                  ChartDataPoint dataPoint =
getLineChartDataPoint(projID, idx, cal);
                  dataSet.addChartDataPoint(dataPoint);
            }

            return ChartDataSetXML.getXML(dataSet);
      }

      /**
       * Gets the line chart data point.
       * The data point contains the number of test execution
       * successes, failures, errors, inconclusive, and
       * not-executed results associated with a given project
       * for a particular number of days.
       *
       * @param projID the TPTeam ID of the project
       * @param daysPrev the number of previous days to include
       * @param Calendar a calendar to help with date math
       * @return the ChartDataPoint
       * @throws Exception
       */
      private static ChartDataPoint getLineChartDataPoint(int projID,
                  int daysPrev, Calendar cal) throws Exception {
            Session s = null;
            Transaction tx = null;
            ChartDataPoint dataPoint = new ChartDataPoint();
            try {

                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();
                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }

                  tx = s.beginTransaction();


      /*************************************************************
***
                   * Oracle Legacy Code
                   *
*************************************************************************
*
                   * String SQL_QUERY = "SELECT te.status, count(*)
FROM TestExecution
                   * as te " + " join te.test as t " + " WHERE
t.project.id = ? AND
                   * t.isFolder = 'N' " + " AND te.execDate = " + "
(SELECT
                   * max(te2.execDate) from TestExecution te2 where
te2.test.id =
```

```
                     * te.test.id " + "AND te2.execDate <= sysdate - ?) "
+ " GROUP BY
                     * te.status";

*****************************************************************/

                    // MySQL Native SQL
                    String SQL_QUERY = "SELECT te.status, count(*) FROM
Test_Execution AS te "
                                 + " INNER JOIN Test as t on te.test_id =
t.id "
                                 + " WHERE t.proj_id = ? AND t.is_Folder =
'N' "
                                 + " AND te.exec_Date = "
                                 + " (SELECT max(te2.exec_Date) FROM
Test_Execution te2 WHERE te2.test_id = te.test_id "
                                 + " AND te2.exec_Date < date_sub(now(),
INTERVAL ? day))"
                                 + " GROUP BY te.status";

                    Query query = s.createSQLQuery(SQL_QUERY);
                    query.setInteger(0, projID);
                    query.setInteger(1, daysPrev);

                    int sum = 0;
                    for (Object row : query.list()) {
                          Object[] cols = (Object[]) row;
                          Character status = (Character) cols[0];
                          java.math.BigInteger count =
(java.math.BigInteger) cols[1];

                          if (status == 'P')
                                dataPoint.setPass(count.intValue());
                          else if (status == 'F')
                                dataPoint.setFail(count.intValue());
                          else if (status == 'E')
                                dataPoint.setError(count.intValue());
                          else if (status == 'I')
                                dataPoint.setInconcl(count.intValue());
                          sum += count.intValue();

                    }


      /*************************************************************
***
                     * Oracle Legacy Code
                     *
*********************************************************************
**************
                     * SQL_QUERY = "SELECT count(*) FROM Test as t " + "
WHERE
                     * t.project.id = ? AND t.isFolder = 'N' AND
t.createdDate <=
                     * sysdate - ?";

*****************************************************************/
```

```
                  SQL_QUERY = "SELECT count(*) FROM Test as t "
                              + " WHERE t.proj_id = ? AND t.is_Folder =
'N' "
                              + "AND t.created_Date < date_sub(now(),
INTERVAL ? day)";

                  query = s.createSQLQuery(SQL_QUERY);
                  query.setInteger(0, projID);
                  query.setInteger(1, daysPrev);

                  for (Object row : query.list()) {
                        java.math.BigInteger totalTests =
(java.math.BigInteger) row;
                        int notExec = totalTests.intValue() - sum;
                        dataPoint.setNotExec(notExec);
                  }
                  cal.add(Calendar.DATE, -daysPrev);
                  dataPoint.setDate(cal.getTime());
                  tx.commit();
            } catch (Exception e) {
                  e.printStackTrace();
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return dataPoint;
      }
}
```

ProjectUtil.java

```
/*****************************************************************
 *
 * File     :     ProjectUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A utility class for TPTeam Project operations
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate;

import java.util.HashSet;
import java.util.Set;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.ProjectXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;

/*****************************************************************
 * File           :     ProjectUtil.java
 *
 * Description    :     A utility class for TPTeam Project operations.
 *                      Performs XML serialization of Project objects
 *                      and look-ups by ECF ID.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class ProjectUtil {

      /**
       * Gets all TPTeam projects associated with a given TPTeam user
       *
       * @param ecfID The ECF ID of the TPTeam user.
       * @return a Set of all associated projects
       * @throws Exception
       */
      public static Set<Project> getProjByECFID(String ecfID) throws
Exception {
            Set<Project> projs = null;
            Session s = null;
            Transaction tx = null;
            try {
                  // If in OSGi envrionment, grab session from plugin
activator
                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
```

```
                                    .getCurrentSession();
                } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                }
                tx = s.beginTransaction();

                String hql = "from TpteamUser as user where
user.ecfId =:ecfID";
                Query query = s.createQuery(hql);
                query.setString("ecfID", ecfID);
                TpteamUser user = (TpteamUser) query.uniqueResult();
                if (user == null)
                        return new HashSet<Project>();
                projs = user.getProjects();
                if (projs == null)
                        return new HashSet<Project>();
                for (Project proj : projs) {
                        proj.initSkeleton();
                }
                tx.commit();
            } catch (Exception e) {
                if (tx != null)
                        tx.rollback();
                throw e;
            }
            return projs;
      }

      /**
       * Gets all TPTeam project/product information stored in the
       * TPTeam database
       *
       * @param tpEvent The TPEvent holding the request information
       * @return an XML String
       * @throws Exception
       */
      public static String getProjProdXML(TPEvent tpEvent) throws
Exception {
            Set<Project> projs =
getProjByECFID(tpEvent.getDictionary().get(
                        TPEvent.FROM));

      /*****************************************************************
*******
             * Create stub Projects
             *
             * Betwixt will call all getter methods objects as part of
its
             * introspection. This will cause non-initialized objects
loaded by
             * Hibernate to throw init errors outside of Hibernate
session. Stub
             * objects allow us to use lazy loading and avoid this
error.

*****************************************************************/
```

```
            HashSet<Project> stubProjs = new HashSet<Project>();
            for (Project proj : projs) {
                  Product stubProd = new
Product(proj.getProduct().getId(), proj
                              .getProduct().getName(), null, null);
                  Project stubProj = new Project(proj.getId(),
stubProd, proj
                              .getName(), proj.getDescription(), null,
null);
                  stubProjs.add(stubProj);
            }
            return ProjectXML.getXML(stubProjs);
      }
}
```

TestExecutionUtil.java

```
/*****************************************************************
 *
 * File     :     TestExecutionUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A utility class for TPTeam TestExecution operations
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate;

import java.util.Date;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestExecution;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;

/*****************************************************************
 * File           :     TestExecutionUtil.java
 *
 * Description    :     A utility class for TPTeam TestExecution
 *                      operations
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TestExecutionUtil {

      /**
       * Inserts a test execution into the TPTeam database
       *
       * @param testID the ID of the test
       * @param tpEvent the TPEvent containing the request
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      public static void insertTestExec(String testID, TPEvent tpEvent)
throws Exception {
            // Use plugin activator if in OSGi runtime
            Session s = null;
            if(Activator.getDefault() != null)
            {
                  s = Activator.getDefault().getHiberSessionFactory()
                  .getCurrentSession();
            }
            else
            {
```

```
                  s =
HibernateUtil.getSessionFactory().getCurrentSession();
            }

            Transaction tx = null;
            try {

                  tx = s.beginTransaction();
                  String hql = "from TpteamUser as user where
user.ecfId =:ecfID";
                  Query query = s.createQuery(hql);
                  query.setString("ecfID",
tpEvent.getDictionary().get(TPEvent.ECFID_KEY));
                  List users = query.list();
                  Test test = (Test) s.load(Test.class,
Integer.parseInt(testID));
                  TestExecution testExec = new TestExecution();
                  testExec.setTest(test);

      testExec.setStatus(tpEvent.getDictionary().get(TPEvent.VERDICT_KE
Y).toUpperCase().charAt(0));
                  testExec.setExecDate(new Date());
                  testExec.setTpteamUser((TpteamUser) users.get(0));
                  s.save(testExec);
                  tpEvent.getDictionary().put(TPEvent.PARENT_ID_KEY,
testID);
                  tpEvent.getDictionary().put(TPEvent.ID_KEY,
TPEntity.EXEC + "_" + String.valueOf(testExec.getId()));
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
      }


}
```

TestUtil.java

```
/*****************************************************************
 *
 * File     :     TestUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A utility class for TPTeam Test operations
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate;

import java.util.Date;
import java.util.List;
import java.util.Set;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestType;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;

/*****************************************************************
 * File           :     TestUtil.java
 *
 * Description    :     A utility class for TPTeam Test operations.
 *                      Performs XML serialization and database
 *                      operations upon Test objects.
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class TestUtil {

      /**
       * Gets a List of all Test objects associated with
       * a given project
       * @param projID the ID of the project
       * @return the List of Tests
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      public static List<Test> getTestByProjID(String projID) throws
Exception {
            List<Test> tests = null;
            Session s = null;
            Transaction tx = null;
            try {
                  // Use plugin activator if in OSGi runtime
```

```
                if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                        .getCurrentSession();
                } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                }
                tx = s.beginTransaction();
                String hql = "from Test as test where test.parent is
null and test.project.id =:projID";
                Query query = s.createQuery(hql);
                query.setString("projID", String.valueOf(projID));
                tests = query.list();

                // Initialize fields of interest
                for (Test test : tests) {
                        test.initSkeleton();
                }

                tx.commit();
            } catch (Exception e) {
                if (tx != null)
                        tx.rollback();
                throw e;
            }
            return tests;
      }

      /**
       * Gets a Test object from the database
       * @param testID the ID of the Test
       * @param includeExec true if test executions should be
       *    included, false otherwise
       * @return the Test object
       * @throws Exception
       */
      public static Test getTestByID(String testID, boolean
includeExec)
                throws Exception {
            Test test = null;
            Session s = null;
            Transaction tx = null;
            try {
                // Use plugin activator if in OSGi runtime
                if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                        .getCurrentSession();
                } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                }
                tx = s.beginTransaction();
                String hql = "from Test as test where test.id
=:testID";
                Query query = s.createQuery(hql);
```

```
                  query.setString("testID", String.valueOf(testID));
                  test = (Test) query.list().get(0);
                  test.initProps(includeExec);
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return test;
      }

      /**
       * Gets the XML String serialization of a project test tree
       *
       * @param tpEvent the TPEvent containing the request
       * @return the XML serialization
       * @throws Exception
       */
      public static String getTestTreeXML(TPEvent tpEvent) throws
Exception {
            List<Test> tests =
getTestByProjID(tpEvent.getDictionary().get(
                        TPEvent.PROJECT_ID_KEY));
            return TestXML.getTPEntityXML(tests,
tpEvent.getDictionary().get(
                        TPEvent.PROJECT_KEY));
      }

      /**
       * Gets a stubbed Test object for update operations
       *
       * @param test the Test object to be stubbed
       * @return a stubbed test object
       */
      public static Test getTestUpdateStub(Test test) {
            Test testStub = new Test();
            testStub.setId(test.getId());
            testStub.setName(test.getName());
            testStub.setDescription(test.getDescription());
            testStub.setIsFolder(testStub.getIsFolder());
            if (test.getJunitTests().size() > 0) {
                  JunitTest junit = ((JunitTest[])
test.getJunitTests().toArray(
                              new JunitTest[0]))[0];
                  JunitTest junitStub = new JunitTest();
                  junitStub.setId(junit.getId());
                  junitStub.setEclipseHome(junit.getEclipseHome());
                  junitStub.setWorkspace(junit.getWorkspace());
                  junitStub.setProject(junit.getProject());
                  junitStub.setTestSuite(junit.getTestSuite());
                  junitStub.setReportDir(junit.getReportDir());

      junitStub.setTptpConnection(junit.getTptpConnection());
                  testStub.addJunitTest(junitStub);
            }
            return testStub;
```

```
	}

	/**
	 * Updates a Test in the datbase
	 *
	 * @param tpEvent the TPEvent containing the request
	 * @throws Exception
	 */
	public static void updateTest(TPEvent tpEvent) throws Exception {
		Session s = null;
		// Use plugin activator if in OSGi runtime
		if (Activator.getDefault() != null) {
			s = Activator.getDefault().getHiberSessionFactory()
					.getCurrentSession();
		} else {
			s =
HibernateUtil.getSessionFactory().getCurrentSession();
		}
		Transaction tx = null;
		try {

			String testXML =
tpEvent.getDictionary().get(TPEvent.TEST_XML_KEY);
			Test testStub = TestXML.getTestFromXML(testXML);

			tx = s.beginTransaction();

			// Get the user who is requesting update
			String hql = "from TpteamUser as user where
user.ecfId =:ecfID";
			Query query = s.createQuery(hql);
			query.setString("ecfID",
tpEvent.getDictionary().get(TPEvent.FROM));
			TpteamUser updateUser = (TpteamUser)
query.list().get(0);

			Test test = (Test) s.load(Test.class,
testStub.getId());
			test.setName(testStub.getName());
			test.setDescription(testStub.getDescription());
			if (test.getIsFolder() == 'N' && test.getJunitTests()
!= null
					&& test.getJunitTests().size() > 0) {
				if (testStub.getJunitTests() != null
						&& testStub.getJunitTests().size()
> 0) {
					JunitTest junit = ((JunitTest[])
test.getJunitTests()
							.toArray(new
JunitTest[0]))[0];
					JunitTest junitStub = ((JunitTest[])
testStub
							.getJunitTests().toArray(new
JunitTest[0]))[0];

	junit.setEclipseHome(junitStub.getEclipseHome());
```

```
	junit.setWorkspace(junitStub.getWorkspace());
                              junit.setProject(junitStub.getProject());

	junit.setTestSuite(junitStub.getTestSuite());

	junit.setReportDir(junitStub.getReportDir());

	junit.setTptpConnection(junitStub.getTptpConnection());
                        }
                  }
                  test.setModifiedBy(updateUser);
                  test.setModifiedDate(new Date());

                  // Collect project member ecfIDs
                  Set<TpteamUser> users =
test.getProject().getTpteamUsers();
                  StringBuilder userECFIDs = new StringBuilder();
                  int idx = 0;
                  for (TpteamUser user : users) {
                        if (idx == 0)
                              userECFIDs.append(user.getEcfId());
                        else
                              userECFIDs.append("/" + user.getEcfId());
                        idx++;
                  }
                  String ECFID =
tpEvent.getDictionary().get(TPEvent.FROM);
                  tpEvent.getDictionary().put(TPEvent.ECFID_KEY,
ECFID);
                  tpEvent.getDictionary().put(TPEvent.FROM,
userECFIDs.toString());
                  tpEvent.getDictionary().put(TPEvent.TEST_NAME_KEY,
test.getName());
                  if (test.getDescription() != null)

	tpEvent.getDictionary().put(TPEvent.TEST_DESC_KEY,
                                    test.getDescription());

                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
	}

	/**
	 * Adds a Test object to the database
	 *
	 * @param tpEvent the TPEvent containing the request
	 * @throws Exception
	 */
	public static void addTest(TPEvent tpEvent) throws Exception {
	      Session s = null;
	      // Use plugin activator if in OSGi runtime
```

```
            if (Activator.getDefault() != null) {
                  s = Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
            } else {
                  s =
HibernateUtil.getSessionFactory().getCurrentSession();
            }
            Transaction tx = null;
            try {

                  String testXML =
tpEvent.getDictionary().get(TPEvent.TEST_XML_KEY);
                  Test testStub = TestXML.getTestFromXML(testXML);

                  tx = s.beginTransaction();

                  Test test = new Test();
                  // First, data common to both folders and tests
                  test.setName(testStub.getName());
                  test.setDescription(testStub.getDescription());
                  test.setIsFolder(testStub.getIsFolder());
                  if (testStub.getParent().getId() > 0) {
                        Test parent = (Test) s.load(Test.class,
testStub.getParent()
                                    .getId());
                        test.setParent(parent);
                  }
                  Project proj = (Project) s.load(Project.class,
testStub
                              .getProject().getId());
                  test.setProject(proj);

                  // Get the user who is requesting update
                  String hql = "from TpteamUser as user where
user.ecfId =:ecfID";
                  Query query = s.createQuery(hql);
                  query.setString("ecfID",
String.valueOf(testStub.getCreatedBy()
                              .getEcfId()));
                  TpteamUser addUser = (TpteamUser)
query.list().get(0);
                  test.setCreatedBy(addUser);
                  test.setCreatedDate(new Date());

                  // Set the test type
                  hql = "from TestType as testType where testType.name
=:name";
                  query = s.createQuery(hql);
                  query.setString("name",
testStub.getTestType().getName());
                  TestType testType = (TestType) query.list().get(0);
                  test.setTestType(testType);

                  // Now, add test definition data
                  if (testStub.getIsFolder() == 'N') {
                        // Need to save now so can associate test
definition with test
```

```
                        // ID in database
                        Integer testID = (Integer) s.save(test);
                        // Next, create JUnit entity, pointing to
parent test
                        JunitTest junitStub = ((JunitTest[])
testStub.getJunitTests()
                                    .toArray(new JunitTest[0]))[0];
                        JunitTest junit = new JunitTest();
                        junit.setId(testID);
                        junit.setTest(test);

      junit.setEclipseHome(junitStub.getEclipseHome());
                        junit.setProject(junitStub.getProject());
                        junit.setWorkspace(junitStub.getWorkspace());
                        junit.setReportDir(junitStub.getReportDir());

      junit.setTptpConnection(junitStub.getTptpConnection());
                        junit.setTestSuite(junitStub.getTestSuite());
                        s.save(junit);
                  }
                  s.saveOrUpdate(test);
                  s.flush();

                  // If parent is null, then is top level folder in
project
                  if (test.getParent() != null)
                        test.setPath(test.getParent().getPath() + "." +
test.getId());
                  else
                        test.setPath(String.valueOf(test.getId()));

                  testStub.setId(test.getId());

                  // Collect project member ecfIDs
                  Set<TpteamUser> users =
test.getProject().getTpteamUsers();
                  StringBuilder userECFIDs = new StringBuilder();
                  int idx = 0;
                  for (TpteamUser user : users) {
                        if (idx == 0)
                              userECFIDs.append(user.getEcfId());
                        else
                              userECFIDs.append("/" + user.getEcfId());
                        idx++;
                  }
                  String ECFID =
tpEvent.getDictionary().get(TPEvent.FROM);
                  tpEvent.getDictionary().put(TPEvent.ECFID_KEY,
ECFID);
                  tpEvent.getDictionary().put(TPEvent.FROM,
userECFIDs.toString());
                  tpEvent.getDictionary().put(TPEvent.ID_KEY,
                              String.valueOf(test.getId()));
                  tpEvent.getDictionary().put(TPEvent.TEST_NAME_KEY,
test.getName());
                  if (test.getDescription() != null)
```

```
            tpEvent.getDictionary().put(TPEvent.TEST_DESC_KEY,
                                        test.getDescription());
                  tpEvent.getDictionary().put(TPEvent.TEST_XML_KEY,
                              TestXML.getXML(testStub));
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http

HttpServiceTracker.java

```
/*********************************************************************
 *
 * File     :     HttpServiceTracker.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Registers HTTP servlets and resources with the
 *                OSGi HTTPService instance
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http;

import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;
import org.osgi.service.http.HttpService;
import org.osgi.util.tracker.ServiceTracker;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddProductEntity
;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddProject;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddProjectEntity
;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddTest;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddTestEntity;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddUserEntity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteProduct
;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteProduct
Entity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteProject
;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteProject
Entity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTest;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTest2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTestEnt
ity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteUser;
```

```
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteUserEnt
ity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.AdminProcessTes
tExec;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.ExecTest;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.ExecTest2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateProduct
;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateProduct
Entity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateProject
;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateProject
2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateProject
Entity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest3;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTestEnt
ity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateUser2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateUserEnt
ity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProduct;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProject;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProject2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest3;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewUser2;
```

```
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.add.AddTestByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.add.AddTestEntityByUs
er;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete.DeleteTest2ByU
ser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete.DeleteTestByUs
er;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete.DeleteTestEnti
tyByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec.ExecTest2ByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec.ExecTestByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec.UserProcessTestE
xec;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateTestByUs
er;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateTestByUs
er2;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateTestByUs
er3;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateTestEnti
tyByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateUserProf
ile;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update.UpdateUserProf
ileEntity;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewProductByUse
r;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewProject2ByUs
er;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewProjectByUse
r;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewTest2ByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewTest3ByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewTestByUser;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view.ViewUserProfile;
```

```
/*****************************************************************
 * File           :     HttpServiceTracker.java
 *
 * Description    :     Registers HTTP servlets and resources with the
 *                      OSGi HTTPService instance
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *****************************************************************/
public class HttpServiceTracker extends ServiceTracker {

      /**
       * Constructor
       * @param context the TPManager plug-in context
       */
      public HttpServiceTracker(BundleContext context) {
            super(context, HttpService.class.getName(), null);
      }

      /**
       * Registers all HTTP servlets and resources of
       * TPManager with the OSGi HTTPService
       *
       * @param reference the HTTPService reference
       * @return the HttpService object
       */
      public Object addingService(ServiceReference reference) {
            HttpService httpService = (HttpService)
context.getService(reference);
            try {
                  // Security HTML Pages
                  httpService.registerResources(
                              "/tpteam/login.html",
"/html/security/login.html", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService.registerResources(
                              "/tpteam/error.html",
"/html/security/error.html", null); //$NON-NLS-1$ //$NON-NLS-2$
                  // Admin HTML Pages
                  httpService.registerResources(
                              "/tpteam/admin/index.html",
"/html/admin/index.html", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/admin/add/addProduct.html",
"/html/admin/add/addProduct.html", null); //$NON-NLS-1$ //$NON-NLS-2$

      /*************************************************************
***
                   * Admin Servlets

*************************************************************/
                  httpService
                              .registerServlet(
```

```
      "/tpteam/admin/add/addProductEntity", new AddProductEntity(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/add/addProject", new AddProject(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/add/addProjectEntity", new AddProjectEntity(),
null, null); //$NON-NLS-1$
                  httpService.registerServlet(
                              "/tpteam/admin/add/addTest", new
AddTest(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/add/addTestEntity", new AddTestEntity(), null,
null); //$NON-NLS-1$
                  httpService.registerServlet(
                              "/tpteam/admin/add/addUser", new
AddUser(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/add/addUserEntity", new AddUserEntity(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateProduct", new UpdateProduct(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateProductEntity", new
UpdateProductEntity(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateProject", new UpdateProject(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateProject2", new UpdateProject2(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateProjectEntity", new
UpdateProjectEntity(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(
```

```
      "/tpteam/admin/update/updateTest", new UpdateTest(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateTest2", new UpdateTest2(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateTest3", new UpdateTest3(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateTestEntity", new UpdateTestEntity(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateUser", new UpdateUser(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateUser2", new UpdateUser2(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/update/updateUserEntity", new UpdateUserEntity(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteProduct", new DeleteProduct(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteProductEntity", new
DeleteProductEntity(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteUser", new DeleteUser(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteUserEntity", new DeleteUserEntity(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(
```

```
      "/tpteam/admin/delete/deleteProject", new DeleteProject(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteProjectEntity", new
DeleteProjectEntity(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteTest", new DeleteTest(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteTest2", new DeleteTest2(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/delete/deleteTestEntity", new DeleteTestEntity(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/view/viewProduct", new ViewProduct(), null, null);
//$NON-NLS-1$
                  httpService.registerServlet(
                              "/tpteam/admin/view/viewUser", new
ViewUser(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/view/viewUser2", new ViewUser2(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/view/viewProject", new ViewProject(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/view/viewProject2", new ViewProject2(), null,
null); //$NON-NLS-1$
                  httpService.registerServlet(
                              "/tpteam/admin/view/viewTest", new
ViewTest(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/view/viewTest2", new ViewTest2(), null, null);
//$NON-NLS-1$
                  httpService
                              .registerServlet(
```

```
      "/tpteam/admin/view/viewTest3", new ViewTest3(), null, null);
//$NON-NLS-1$
                  httpService.registerServlet(
                              "/tpteam/admin/exec/execTest", new
ExecTest(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/exec/execTest2", new ExecTest2(), null, null);
//$NON-NLS-1$

      /*************************************************************
***
                   * User Servlets

*************************************************************/
                  httpService
                              .registerServlet(
                                          "/tpteam/user/add/addTest",
new AddTestByUser(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/add/addTestEntity", new AddTestEntityByUser(),
null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateTest", new UpdateTestByUser(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateTest2", new UpdateTestByUser2(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateTest3", new UpdateTestByUser3(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateTestEntity", new
UpdateTestEntityByUser(), null, null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateUser", new UpdateUserProfile(), null,
null); //$NON-NLS-1$
                  httpService
                              .registerServlet(

      "/tpteam/user/update/updateUserEntity", new
UpdateUserProfileEntity(), null, null); //$NON-NLS-1$
                  httpService
```

```
                                .registerServlet(

      "/tpteam/user/delete/deleteTest", new DeleteTestByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/delete/deleteTest2", new DeleteTest2ByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/delete/deleteTestEntity", new
DeleteTestEntityByUser(), null, null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/view/viewProduct", new ViewProductByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(
                                            "/tpteam/user/view/viewUser",
new ViewUserProfile(), null, null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/view/viewProject", new ViewProjectByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/view/viewProject2", new ViewProject2ByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(
                                            "/tpteam/user/view/viewTest",
new ViewTestByUser(), null, null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/view/viewTest2", new ViewTest2ByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/view/viewTest3", new ViewTest3ByUser(), null,
null); //$NON-NLS-1$
                    httpService
                                .registerServlet(
                                            "/tpteam/user/exec/execTest",
new ExecTestByUser(), null, null); //$NON-NLS-1$
                    httpService
                                .registerServlet(

      "/tpteam/user/exec/execTest2", new ExecTest2ByUser(), null,
null); //$NON-NLS-1$
                    httpService
```

```
                              .registerServlet(

      "/tpteam/user/exec/processTestExec", new UserProcessTestExec(),
null, null); //$NON-NLS-1$

                  // User HTML Pages
                  httpService.registerResources(
                              "/tpteam/user/index.html",
"/html/user/index.html", null); //$NON-NLS-1$ //$NON-NLS-2$
                  // Images
                  httpService.registerResources(
                              "/tpteam/images/open.gif",
"/html/images/open.gif", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(
                                          "/tpteam/images/closed.gif",
"/html/images/closed.gif", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService.registerResources(
                              "/tpteam/images/doc.gif",
"/html/images/doc.gif", null); //$NON-NLS-1$ //$NON-NLS-2$

      /*************************************************************
***
                   * Scripts

*************************************************************/
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_test.js", "/html/scripts/add_test.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_test_tree.css",
"/html/scripts/add_test_tree.css", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_test_tree.js",
"/html/scripts/add_test_tree.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_test_type.js",
"/html/scripts/add_test_type.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_test_folder.js",
"/html/scripts/add_test_folder.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/add_user.js", "/html/scripts/add_user.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
```

```
                              .registerResources(

      "/tpteam/scripts/update_prod.js", "/html/scripts/update_prod.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/update_proj.js", "/html/scripts/update_proj.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/update_test_tree.js",
"/html/scripts/update_test_tree.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/update_test_folder.js",
"/html/scripts/update_test_folder.js", null); //$NON-NLS-1$ //$NON-NLS-
2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/update_test_def.js",
"/html/scripts/update_test_def.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/update_user.js", "/html/scripts/update_user.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/delete_prod.js", "/html/scripts/delete_prod.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/delete_user.js", "/html/scripts/delete_user.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/delete_proj.js", "/html/scripts/delete_proj.js",
null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/delete_test_tree.js",
"/html/scripts/delete_test_tree.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(

      "/tpteam/scripts/view_test_tree.js",
"/html/scripts/view_test_tree.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  httpService
                              .registerResources(
```

```
      "/tpteam/scripts/exec_test_tree.js",
"/html/scripts/exec_test_tree.js", null); //$NON-NLS-1$ //$NON-NLS-2$
                  // TPTP Resources
                  //
httpService.registerServlet("/tpteam/processTestExec", new
                  // ProcessTestExecServlet(), null, null); //$NON-NLS-
1$
                  httpService
                              .registerServlet(

      "/tpteam/admin/exec/processTestExec", new AdminProcessTestExec(),
null, null); //$NON-NLS-1$
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return httpService;
      }

      /**
       * HttpService cleanup
       */
      public void removedService(ServiceReference reference, Object
service) {
            HttpService httpService = (HttpService) service;
            httpService.unregister("/index.html"); //$NON-NLS-1$
            httpService.unregister("/processTestExec"); //$NON-NLS-1$
            super.removedService(reference, service);
      }
}
```

ServletUtil.java

```
/**********************************************************************
 *
 * File     :     ServletUtil.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A utility class for rendering HTML
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http;

import java.io.IOException;
import java.io.PrintWriter;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Stack;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;

/**********************************************************************
 * File           :     ServletUtil.java
 *
 * Description    :     A utility class for rendering HTML
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class ServletUtil extends HttpServlet {

      private static final long serialVersionUID = 1L;

      // JavaScript and Custom Style Sheet convenience Strings
      public static final String ADD_TEST_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/add_test.js\">\n";

      public static final String ADD_TEST_TREE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/add_test_tree.js\">\n";

      public static final String ADD_TEST_TYPE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/add_test_type.js\">\n";
```

```
      public static final String ADD_TEST_FOLDER_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/add_test_folder.js\">\n";

      public static final String ADD_USER_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/add_user.js\">\n";

      public static final String UPDATE_PROD_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_prod.js\">\n";

      public static final String UPDATE_PROJ_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_proj.js\">\n";

      public static final String UPDATE_TEST_TREE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_test_tree.js\">\n";

      public static final String UPDATE_TEST_FOLDER_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_test_folder.js\">\n";

      public static final String UPDATE_TEST_DEF_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_test_def.js\">\n";

      public static final String UPDATE_USER_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/update_user.js\">\n";

      public static final String ADD_TEST_TREE_CSS = "<link
href=\"/bridge/tpteam/scripts/add_test_tree.css\" rel=\"stylesheet\"
type=\"text/css\">\n";

      public static final String DELETE_PROD_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/delete_prod.js\">\n";

      public static final String DELETE_USER_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/delete_user.js\">\n";

      public static final String DELETE_PROJ_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/delete_proj.js\">\n";

      public static final String DELETE_TEST_TREE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/delete_test_tree.js\">\n";

      public static final String VIEW_TEST_TREE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/view_test_tree.js\">\n";
```

```
      public static final String EXEC_TEST_TREE_JS = "<script
type=\"text/javascript\" language=\"JavaScript\"
src=\"/bridge/tpteam/scripts/exec_test_tree.js\">\n";

      /** Timestamp data format to be used in TPManager operations */
      public static final SimpleDateFormat TIMESTAMP_FORMAT = new
SimpleDateFormat(
                  "dd-MMM-yyyy HH:mm:ss.SSS zzz");

      /**
       * Provides the HTML header for all TPTeam administrative
       * Web pages
       * @param request the servlet request object
       * @param response the servlet response object
       * @param javaScript JavaScript string to be included
       * @throws ServletException
       * @throws IOException
       */
      public void adminHeader(HttpServletRequest request,
                  HttpServletResponse response, String javaScript)
                  throws ServletException, IOException {
            if (javaScript == null)
                  javaScript = "";
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String header = "<html><head><title>TPTeam Admin
Pages</title>\n"
                        + javaScript
                        + "</head>\n"
                        + "<body><div align=\"center\"><h3>TPTeam Admin
Pages</h3>\n"
                        + "<a
href=\"/bridge/tpteam/admin/index.html\">Home</a><hr size=\"2\">\n";
            out.println(header);
      }

      /**
       * Provides the HTML header for all TPTeam user
       * Web pages
       * @param request the servlet request object
       * @param response the servlet response object
       * @param javaScript JavaScript string to be included
       * @throws ServletException
       * @throws IOException
       */
      public void userHeader(HttpServletRequest request,
                  HttpServletResponse response, String javaScript)
                  throws ServletException, IOException {
            if (javaScript == null)
                  javaScript = "";
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String header = "<html><head><title>TPTeam Pages</title>\n"
                        + javaScript
                        + "</head>\n"
                        + "<body><div
align=\"center\"><h3>TPTeam</h3>\n"
```

```
                        + "<a
href=\"/bridge/tpteam/user/index.html\">Home</a><hr size=\"2\">\n";
            out.println(header);
      }

      /**
       * Provides the HTML footer for all TPTeam administrative
       * Web pages
       * @param request the servlet request object
       * @param response the servlet response object
       * @throws ServletException
       * @throws IOException
       */
      public void adminFooter(HttpServletRequest request,
                  HttpServletResponse response) throws
ServletException, IOException {
            PrintWriter out = response.getWriter();
            String footer = "</div><hr size=\"2\"></body></html>\n";
            out.println(footer);
            out.close();
      }

      /**
       * Provides the HTML footer for all TPTeam user
       * Web pages
       * @param request the servlet request object
       * @param response the servlet response object
       * @throws ServletException
       * @throws IOException
       */
      public void userFooter(HttpServletRequest request,
                  HttpServletResponse response) throws
ServletException, IOException {
            adminFooter(request, response);
      }

      /**
       * Provides the HTML reply (excluding header & footer)
       * for all TPTeam administrative Web pages
       *
       * @param request the servlet request object
       * @param response the servlet response object
       * @throws ServletException
       * @throws IOException
       */
      public void adminReply(HttpServletRequest request,
                  HttpServletResponse response, String reply)
                  throws ServletException, IOException {
            PrintWriter out = response.getWriter();
            out.println(reply);
      }

      /**
       * Provides the HTML reply (excluding header & footer)
       * for all TPTeam user Web pages
       *
       * @param request the servlet request object
```

```
     * @param response the servlet response object
     * @throws ServletException
     * @throws IOException
     */
    public void userReply(HttpServletRequest request,
                HttpServletResponse response, String reply)
                throws ServletException, IOException {
          adminReply(request, response, reply);
    }

    /**
     * Provides the HTML error reply, excluding
     * header and footer, for all TPTeam
     * administrative Web pages
     *
     * @param request the servlet request object
     * @param response the servlet response object
     * @throws ServletException
     * @throws IOException
     */
    public void adminError(HttpServletRequest request,
                HttpServletResponse response, String error)
                throws ServletException, IOException {
          adminHeader(request, response, null);
          error += "<p/><a href=\"javascript:history.back(1)\">Try
Again</a><p/>";
          adminReply(request, response, error);
          adminFooter(request, response);
    }

    /**
     * Provides the HTML error reply, excluding
     * header and footer, for all TPTeam
     * user Web pages
     *
     * @param request the servlet request object
     * @param response the servlet response object
     * @throws ServletException
     * @throws IOException
     */
    public void userError(HttpServletRequest request,
                HttpServletResponse response, String error)
                throws ServletException, IOException {
          userHeader(request, response, null);
          error += "<p/><a href=\"javascript:history.back(1)\">Try
Again</a><p/>";
          userReply(request, response, error);
          userFooter(request, response);
    }

    /**
     * Rendors an HTML error message
     * @param request the servlet request object
     * @param response the servlet response object
     * @param error the error message
     * @param servlet the servlet throwing the error
     * @throws ServletException
```

```
       * @throws IOException
       */
      protected void throwError(HttpServletRequest req,
HttpServletResponse resp,
                  StringBuffer error, HttpServlet servlet) throws
ServletException,
                  IOException {
            if (servlet instanceof UserServlet)
                  userError(req, resp, error.toString());
            else
                  adminError(req, resp, error.toString());
      }

      /**
       * Rendors an entire HTML page
       *
       * @param request the servlet request object
       * @param response the servlet response object
       * @param javaScript the Web page JavaScript
       * @param servlet the servlet throwing the error
       * @throws ServletException
       * @throws IOException
       */
      protected void showPage(HttpServletRequest req,
HttpServletResponse resp,
                  StringBuffer reply, String javaScript, HttpServlet
servlet)
                  throws ServletException, IOException, Exception {
            if (servlet instanceof UserServlet) {
                  userHeader(req, resp, javaScript);
                  userReply(req, resp, reply.toString());
                  userFooter(req, resp);
            } else {
                  adminHeader(req, resp, javaScript);
                  adminReply(req, resp, reply.toString());
                  adminFooter(req, resp);

            }
      }

      /**
       * Gets the JavaScript for rendering a test tree
       * @return the JavaScript String
       */
      public String getTreeJavaScript() {
            String javaScript = ""
                        + " var openImg = new Image();\n"
                        + " openImg.src = \"open.gif\";\n"
                        + " var closedImg = new Image();\n"
                        + " closedImg.src = \"closed.gif\";\n"
                        + " var selectedID = \"\";\n"
                        +

                        " function showBranch(branch){\n"
                        + "\t var objBranch =
document.getElementById(branch).style;\n"
                        + "\t if(objBranch.display==\"block\")\n"
```

```
                        + "\t\t objBranch.display=\"none\";\n"
                        + "\t else\n"
                        + "\t {\n"
                        + "\t\t objBranch.display=\"block\";\n"
                        + "\t }\n"
                        + "}\n"
                        +

                        "function swapFolder(img){\n"
                        + "\t objImg = document.getElementById(img);\n"
                        + "\t if(objImg.src.indexOf('closed.gif')>-
1)\n"
                        + "\t\t objImg.src = openImg.src;\n"
                        + "\t else\n"
                        + "\t\t objImg.src = closedImg.src;\n"
                        + "}\n"
                        +

                        "function makeBold(id){\n"
                        + "\t if(selectedID != \"\")\n"
                        + "\t {\n"
                        + "\t\t
document.getElementById(selectedID).style.fontWeight = 'normal';\n"
                        + "\t}\n"
                        + "\t
document.getElementById(id).style.fontWeight = 'bold';\n"
                        + "\tselectedID = id;\n" + "}\n";
            return javaScript;
      }

      /**
       * Gets all top-level test folders for
       * a given project
       *
       * @param projID the ID of the project
       * @return the HTML String of the folders
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      public static String getTestTreeFolders(String projID) throws
Exception {

            Session s = null;
            if (Activator.getDefault() != null) {
                  s = Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
            } else {
                  // For standalone debug
                  s =
HibernateUtil.getSessionFactory().getCurrentSession();
            }
            Transaction tx = null;
            StringBuffer tree = new StringBuffer();

            try {
                  tx = s.beginTransaction();
```

```
                  // Order by desc since we're pushing on stack: alpha
first on top
                  List<Test> tests = s
                              .createQuery(
                                          "from Test as test where
test.project.id = "
                                                      + projID
                                                      + " and
test.parent is null order by test.path desc")
                              .list();

                  tree.append(getRootHeader());
                  tree.append(getTree(tests, true));
                  tree.append("</SPAN>");

                  tx.commit();

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }

            return tree.toString();
      }

      /**
       * Extracts the depth of a test tree node
       * from the path
       *
       * @param path the path
       * @return the depth
       */
      public static int getDepth(String path) {
            String pathNoDots = path.replaceAll("\\.", "");
            return path.length() - pathNoDots.length();
      }

      /**
       * Gets the String representation of a test tree
       * @param tests the List of tests
       * @param onlyFolders true if only display folders, false
otherwise
       * @return the test tree
       */
      public static String getTree(List<Test> tests, boolean
onlyFolders) {
            Stack<Test> nodes = new Stack<Test>();
            nodes.addAll(tests);
            // Initialize node level to top level
            int currLevel = 0;
            StringBuffer tree = new StringBuffer();

            /**
             * While node stack is not empty, add header String, push
child nodes
             * repeat
```

```
             */
            while (!nodes.empty()) {
                  Test test = nodes.pop();

                  /**
                   * If depth in tree decreased, print closing </SPAN>
tag for
                   * previous parent
                   */
                  int depth = getDepth(test.getPath());
                  if (depth < currLevel)
                        tree.append(getNodeFooter(depth, currLevel));
                  currLevel = depth;

                  tree.append(getNodeHeader(test, depth, onlyFolders));

                  // push children onto stack
                  if (test.getIsFolder() == 'Y' &&
test.getChildren().size() > 0)
                        nodes.addAll(test.getChildren());
                  else if (test.getIsFolder() == 'Y')
                        tree.append(getNodeFooter(depth));
            }
            // Add final closing </SPAN> tag
            tree.append(getNodeFooter(0));
            return tree.toString();
      }

      private static String getNodeHeader(Test test, int depth,
                  boolean onlyFolders) {
            StringBuffer header = new StringBuffer();
            String testId = String.valueOf(test.getId());
            String testName = test.getName();
            if (test.getIsFolder() == 'Y') {
                  header.append(getPad(depth));
                  header
                              .append("<div class=\"trigger\"
onClick=\"showBranch('branch_"
                                          + testId
                                          + "');swapFolder('folder_"
                                          + testId
                                          + "');makeBold('"
                                          + testId
                                          + "');\" id=\""
                                          + testId
                                          + "\">\n");
                  header
                              .append("<img
src=\"/bridge/tpteam/images/closed.gif\" border=\"0\" id=\"folder_"
                                          + testId + "\">\n");
                  header.append(test.getName() + "\n");
                  header.append("</div>\n");
                  header.append("<span class=\"branch\" id=\"branch_" +
testId
                              + "\">\n");
            } else if (onlyFolders == false) {
                  header.append(getPad(depth)
```

```
                              + "<img
src=\"/bridge/tpteam/images/doc.gif\"> "
                              + "<a href=\"#\" id=\"" + testId
                              + "\" onClick=\"makeBold('" + testId
                              + "'); return false;\">" + testName +
"</a><br>\n");
            }
            return header.toString();
      }

      /**
       * Helper method to get the test tree header
       * @return the header
       */
      private static String getRootHeader() {
            StringBuffer header = new StringBuffer();
            header
                        .append("<div class=\"trigger\"
onClick=\"showBranch('branch_0');");
            header.append("swapFolder('folder_0');makeBold('0');\"
id=\"0\">\n");
            header
                        .append("<img
src=\"/bridge/tpteam/images/closed.gif\" border=\"0\"
id=\"folder_0\">\n");
            header.append("Project Root\n");
            header.append("</div>\n");
            header.append("<span class=\"branch\" id=\"branch_0\">\n");
            return header.toString();
      }

      /**
       * Helper method to get a test tree node footer
       *
       * @param depth the node depth
       * @param currLevel the current level in the tree
       * @return the footer
       */
      private static String getNodeFooter(int depth, int currLevel) {
            StringBuffer footer = new StringBuffer();
            while (currLevel > depth)
                  footer.append(getPad(--currLevel) + "</SPAN>\n");
            return footer.toString();
      }

      /**
       * Helper method to get the test tree node footer
       *
       * @param depth the depth of the node in the tree
       * @return the footer
       */
      private static String getNodeFooter(int depth) {
            return getPad(depth) + "</SPAN>\n";
      }

      /**
       * Helper method to get the node footer padding
```

```
	 *
	 * @param depth the node depth
	 * @return the padding
	 */
	private static String getPad(int depth) {
		StringBuffer pad = new StringBuffer();
		for (int idx = 0; idx < depth; idx++)
			pad.append("\t");
		return pad.toString();
	}

	/**
	 * Gets the hash of a plaintext String using
	 * the SHA1 algorithm
	 *
	 * @param plainText the plain text
	 * @return the hash
	 * @throws NoSuchAlgorithmException
	 */
	public static String getSHA1Hash(String plainText)
			throws NoSuchAlgorithmException {
		MessageDigest md = MessageDigest.getInstance("SHA1");
		md.update(plainText.getBytes());
		byte[] mdbytes = md.digest();
		StringBuffer hexString = new StringBuffer();
		for (int i = 0; i < mdbytes.length; i++) {
			String hex = Integer.toHexString(0xFF & mdbytes[i]);
			if (hex.length() == 1)
				hexString.append('0');

			hexString.append(hex);
		}
		return hexString.toString().toUpperCase();
	}

	/**
	 * Gets the String representation of a project's test
	 * tree
	 *
	 * @param projID the ID of the project
	 * @param onlyFolders true if include only folders,
	 *    false otherwise
	 * @return the test tree String
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	public static String getTestTree(String projID, boolean
onlyFolders)
			throws Exception {

		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();

		// For standalone debug
		// Session s =
HibernateUtil.getSessionFactory().getCurrentSession();
		Transaction tx = null;
```

```
            StringBuffer tree = new StringBuffer();

            try {
                  tx = s.beginTransaction();

                  // Order by desc since we're pushing on stack: alpha
first on top

                  List<Test> tests = s
                              .createQuery(
                                          "from Test as test where
test.project.id = "
                                                      + projID
                                                      + " and
test.parent is null order by test.path desc")
                              .list();

                  tree.append(getRootHeader());
                  tree.append(getTree(tests, onlyFolders));
                  tree.append("</SPAN>");

                  tx.commit();

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return tree.toString();
      }

      /**
       * Gets the TPTeam user ID of the servlet's
       * remote user
       *
       * @param userName the servlet remote user's name
       * @return the TPTeam ID of the user
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      public static int getRemoteUserID(String userName) throws
Exception {

            Session s = null;
            if (Activator.getDefault() != null) {
                  s = Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
            } else {
                  s =
HibernateUtil.getSessionFactory().getCurrentSession();
            }
            Transaction tx = null;
            int userId;
            try {
                  tx = s.beginTransaction();
                  List<TpteamUser> users = s.createQuery(
```

```
                              "from TpteamUser as user where
user.userName = '"
                                          + userName + "'").list();
                  userId = users.get(0).getId();
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return userId;
      }

}
```

UserServlet.java

```
/**********************************************************************
 *
 * File     :     UserServlet.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     An interface used to identify non-administrative
 *                Web page servlets
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http;

/**********************************************************************
 * File           :     UserServlet.java
 *
 * Description    :     An interface used to identify non-
 *                      administrative Web page servlets
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public interface UserServlet {

}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add

AddProductEntity.java

```
/**********************************************************************
 *
 * File     :     AddProductEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet used for adding new Products to the TPTeam
 *                database
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     AddProductEntity.java
 *
 * Description    :     Servlet used for adding new Products to the
 *                      TPTeam database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class AddProductEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Creates a new Product based on input parameters
       * and persists it to the database
       *
       * Displays results, including errors to the user
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {

            String  name = req.getParameter("prodName");
            String description = req.getParameter("prodDesc");

            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {
                  tx = s.beginTransaction();
                  Product prod = new Product();
                  prod.setName(name);
                  prod.setDescription(description);
                  s.save(prod);
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                  e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
                  adminHeader(req, resp, null);
                  String reply = "<h3>Add Product " + name + " was
Successful</h3>";
                  adminReply(req, resp, reply);
                  adminFooter(req, resp);
      }


}
```

AddProject.java

```
/*********************************************************************
 *
 * File     :     AddProject.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for creating
 *                a new TPTeam Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     AddProject.java
 *
 * Description    :     Servlet that displays an input form for
 *                      creating a new TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class AddProject extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;
      private boolean mIsProdAvailable = false;
      private boolean mIsTeamUserAvailable = false;
      private String mProdOptions = null;
      private String mTeamOptions = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Renders the new Project input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try
            {
            getProdOptions();
            getTeamOptions();
            if(mIsProdAvailable == false || mIsTeamUserAvailable ==
false)
            {
                  throwError(req, resp);
            }
            else
            {
                  showPage(req, resp);
            }
            }
             catch (Exception e) {
                        String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                        e.getCause() + "</h3>";
                        adminError(req, resp, error);
                        return;
                  }
      }

      /**
       * Helper method that gets all Products and wraps
       * them in HTML option tags
       *
       * @return The Product option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getProdOptions() throws Exception
      {
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<Product> prods = null;
            mProdOptions = "";
            try {

                  tx = s.beginTransaction();
                  prods = s.createQuery("from Product as p order by
p.name asc").list();
                  for(Product prod : prods)
                  {
                        mProdOptions += "<option value=\"" +
prod.getId() + "\">" + prod.getName() + "</option>";
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
```

```
            }
            if(!mProdOptions.equals(""))
                  mIsProdAvailable = true;

            mProdOptions = "<option selected>Choose Product</option>" +
mProdOptions;

            return mProdOptions;
      }

      /**
       * Helper method that gets all TPTeam users and
       * wraps them into HTML option tags
       *
       * @return The TPTeam user option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getTeamOptions() throws Exception
      {
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<TpteamUser> users = null;
            mTeamOptions = "";
            try {

                  tx = s.beginTransaction();

                  users = s.createQuery("from TpteamUser as team order
by team.lastName asc").list();
                  for(TpteamUser user : users)
                  {
                        mTeamOptions += "<option value=\"" +
user.getId() + "\">" + user.getLastName() + ", " + user.getFirstName()+
"</option>";
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(!mTeamOptions.equals(""))
                  mIsTeamUserAvailable = true;

            return mTeamOptions;
      }

      /**
       * Helper method to render errors as HTML
       * @param req The Servlet Request
       * @param resp The Servlet Response
       *
       * @throws ServletException
       * @throws IOException
```

```
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
      throws ServletException, IOException
      {
            String error = "<h3>Error: No Product or TPTeam Member
Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method to render the HTML page, including
       * JavaScript
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
      throws ServletException, IOException
      {
            String javaScript =
            "<script language=\"JavaScript\">\n<!--\n" +
                  "function validateForm ( )\n" +
                  "{\n var name = document.addProj.projName.value;\n" +
                        "var desc = document.addProj.projDesc.value;\n"
+
                        "var prod = document.addProj.prod;\n" +
                        "if( name == \"\")\n" +
                        "{\n alert(\"Please enter at least a project
name.\");\n" +
                              "return false;\n}\n" +
                     "else if(prod.selectedIndex == 0)\n" +
                     "{\n alert(\"Please select a product.\");\n" +
                        "return false;\n}\n" +
                        "document.addProj.action =
\"addProjectEntity\";\n" +
                        "document.addProj.submit();\n" +
                   "return true;\n}\n-->\n</script>\n";

            String reply = "<h4>Add Project</h4>\n";
            reply += "<form name=\"addProj\" method=\"post\"
onSubmit=\"return validateForm( );\">\n";
            reply += "<table ><tr><th>Name:</th><td><input
type=\"text\" name=\"projName\" size=\"25\"></td></tr>\n";
            reply += "<tr><th>Description:</th><td><input type=\"text\"
name=\"projDesc\" size=\"50\"></td></tr>\n";
            reply += "<tr><th>Product:</th><td><select name=\"prod\">"
+ mProdOptions + "</select></td></tr>\n";
            reply += "<tr><th>Team Members:</th><td><select multiple
size=\"5\" name=\"team\">" + mTeamOptions + "</select></td></tr>\n";
            reply += "</table>\n<br>\n<input type=\"submit\"
value=\"Add\">\n</form>\n";

            adminHeader(req, resp, javaScript);
```

```
                adminReply(req, resp, reply);
                adminFooter(req, resp);
        }


}
```

AddProjectEntity.java

```
/*********************************************************************
 *
 * File     :     AddProjectEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that creates a new Project and persists
 *                it to the TPTeam database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     AddProjectEntity.java
 *
 * Description    :     Servlet that creates a new Project and persists
 *                      it to the TPTeam database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class AddProjectEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Creates an new Project from input parameters, persists
       * it to the database.  Renders results, including errors,
       * to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            String name = req.getParameter("projName");
            String description = req.getParameter("projDesc");
            String prodId = req.getParameter("prod");
            String[] userIds = req.getParameterValues("team");
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {

                  tx = s.beginTransaction();
                  Project proj = new Project();
                  proj.setName(name);
                  proj.setDescription(description);
                  Product prod = (Product) s.load(Product.class, new
Integer(prodId));
                  proj.setProduct(prod);
                  for (String userId : userIds) {
                        TpteamUser tpTeamUser = (TpteamUser)
s.load(TpteamUser.class,
                                    new Integer(userId));
                        // Project is not the inverse=true, so use it
to associate
                        // proj_user map table
                        proj.addToTpteamUsers(tpTeamUser);
                  }
                  s.save(proj);
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
            adminHeader(req, resp, null);
            String reply = "<h3>Add Project " + name + " was
Successful</h3>";
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

AddTest.java

```
/*********************************************************************
 *
 * File     :     AddTest.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays input forms for creating
 *                a new TPTeam Test
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestType;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     AddTest.java
 *
 * Description    :     Servlet that displays input forms for creating
 *                      a new TPTeam Test
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class AddTest extends ServletUtil {

      protected static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProjAvailable = false;

      protected String mProjOptions = null;

      protected String mTestTypeOptions = null;

      protected String mTestName = null;

      protected String mTestDesc = null;

      protected String mTestTypeIDName = null;

      protected String mTestTypeID = null;

      protected String mTestTypeName = null;
```

```
      protected String mProjIDName = null;

      protected String mProjID = null;

      protected String mProjName = null;

      protected String mRemoteUser = null;

      // Default values for JUnit tests

      public static final String ECLIPSE_HOME =
"c:/Java/Eclipse3.2.1/eclipse";

      public static final String ECLIPSE_WORKSPACE =
"c:/workspace_tpteam_test";

      public static final String ECLIPSE_PROJECT =
"edu.harvard.fas.rbrady.tpteam.test";

      public static final String REPORT_DIR = "c:/tpteam/test/reports";

      public static final String TPTP_CONN =
"tptp:rac://localhost:10002/default";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Renders the appropriate form for a new Test, folder or
       * definiton.  Displays any errors to the user.
       *
       * @param req the Servlet request
       * @param resp the Servlet response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mTestName = req.getParameter("testName");
                  mTestDesc = req.getParameter("testDesc");

      /*************************************************************
***
                   * test type and project are single select lists so
getParameter
                   * gets the single value.

*************************************************************/
                  mTestTypeIDName = req.getParameter("testType");
                  mProjIDName = req.getParameter("proj");
                  mRemoteUser = req.getRemoteUser();

                  if (mTestName != null && mTestTypeIDName != null
```

```
                        && mProjIDName != null) {
                    getPage2(req, resp);
                } else {
                    getPage1(req, resp);
                }
            } catch (Exception e) {
                StringBuffer error = new StringBuffer("<h3>Error: " +
e.getMessage() + "<br>"
                        + e.getCause() + "</h3>");
                throwError(req, resp, error, this);
                return;
            }
      }

      /**
       * Renders the core data input form for a new Test:
       * name, description, and type.
       *
       * @param req the Servlet Request
       * @param resp the Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      public void getPage1(HttpServletRequest req, HttpServletResponse
resp)
                throws ServletException, IOException, Exception {
            getProjOptions();
            getTestTypeOptions();
            if (mIsProjAvailable == false) {
                StringBuffer error = new StringBuffer(
                        "<h3>Error: No Project Available.  A
Project needs to be created first.</h3>");
                throwError(req, resp, error, this);
            } else {
                showAddTestPage1(req, resp);
            }
      }

      /**
       * Renders the detailed form input for a new Test:
       * the parent folder and any test definition details.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      public void getPage2(HttpServletRequest req, HttpServletResponse
resp)
                throws ServletException, IOException, Exception {
            // Split out IDs and Names from form parameters
            mTestTypeID = mTestTypeIDName.split(":")[0];
            mTestTypeName = mTestTypeIDName.replaceFirst(mTestTypeID +
":", "");
            mProjID = mProjIDName.split(":")[0];
```

```
			mProjName = mProjIDName.replaceFirst(mProjID + ":", "");
			showAddTestPage2(req, resp);
		}


		/**
		 * Gets all TPTeam Projects and wraps them
		 * into HTML select option tags
		 *
		 * @return The Project select option tags
		 * @throws Exception
		 */
		@SuppressWarnings("unchecked")
		protected String getProjOptions() throws Exception {
			Session s = Activator.getDefault().getHiberSessionFactory()
					.getCurrentSession();

			Transaction tx = null;
			List<Project> projs = null;
			mProjOptions = "";
			try {

				tx = s.beginTransaction();

				projs = s.createQuery("from Project as p order by
p.name asc")
						.list();

				// Concatenate id with name to save database call
later.
				for (Project proj : projs) {
					mProjOptions += "<option value=\"" +
proj.getId() + ":"
							+ proj.getName() + "\">" +
proj.getName() + "</option>";
				}

				tx.commit();
			} catch (Exception e) {
				if (tx != null)
					tx.rollback();
				throw e;
			}
			if (!mProjOptions.equals(""))
				mIsProjAvailable = true;

			mProjOptions = "<option selected>Choose Project</option>"
					+ mProjOptions;

			return mProjOptions;
		}

		/**
		 * Gets all TPTeam TestTypes and wraps them
		 * into HTML select option tags
		 *
		 * @return The TestType select option tags
```

```
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getTestTypeOptions() throws Exception {
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		Transaction tx = null;
		List<TestType> types = null;
		mTestTypeOptions = "";
		try {

			tx = s.beginTransaction();

			types = s
					.createQuery(
							"from TestType as type where
type.id > 0 order by type.name asc")
					.list();
			// Concatenate id with name to save database call
later.
			for (TestType type : types) {
				mTestTypeOptions += "<option value=\"" +
type.getId() + ":"
						+ type.getName() + "\">" +
type.getName() + "</option>";
			}

			tx.commit();
		} catch (Exception e) {
			if (tx != null)
				tx.rollback();
			throw e;
		}
		mTestTypeOptions = "<option selected>Choose Test
Type</option>"
				+ mTestTypeOptions;

		return mTestTypeOptions;
	}

	/**
	 * Helper method for rendering core input data form
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws ServletException
	 * @throws IOException
	 * @throws Exception
	 */
	protected void showAddTestPage1(HttpServletRequest req,
			HttpServletResponse resp) throws ServletException,
IOException,
			Exception {

		StringBuffer reply = new StringBuffer();
		reply.append("<h4>Add Test Plan Object, Page 1 of
2</h4>\n");
		reply
```

```
                        .append("<form name=\"addTestPage1\"
method=\"post\" onSubmit=\"return validatePage1Form( );\">\n");
            reply
                        .append("<table ><tr><th>Name:</th><td><input
type=\"text\" name=\"testName\" size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>Description:</th><td><input
type=\"text\" name=\"testDesc\" size=\"50\"></td></tr>\n");
            reply
                        .append("<tr><th>Test Plan Object
Type:</th><td><select name=\"testType\">"
                                    + mTestTypeOptions +
"</select></td></tr>\n");
            reply.append("<tr><th>Project:</th><td><select
name=\"proj\">"
                        + mProjOptions + "</select></td></tr>\n");
            reply
                        .append("</table>\n<br>\n<input type=\"submit\"
value=\"Add\">\n</form>\n");

            showPage(req, resp, reply, ServletUtil.ADD_TEST_JS, this);
      }

      /**
       * Helper method for rendering the detailed
       * input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      protected void showAddTestPage2(HttpServletRequest req,
                  HttpServletResponse resp) throws ServletException,
IOException,
                  Exception {
            StringBuffer reply = new StringBuffer();
            reply.append("<h4>Add Test Plan Element, Page 2 of
2</h4>\n");
            reply
                        .append("<form name=\"addTestPage2\"
method=\"post\" onSubmit=\"return validatePage2Form( );\">\n");
            reply.append("<table
align=\"center\"><tr><th>Name:</th><td>"
                        + mTestName);
            reply.append("<input type=\"hidden\" name=\"testName\"
value=\""
                        + mTestName + "\"></td></tr>\n");
            reply.append("<tr><th>Description:</th><td>" + mTestDesc
                        + "<input type=\"hidden\" name=\"testDesc\"
value=\""
                        + mTestDesc + "\"></td></tr>\n");
            reply.append("<tr><th>Test Type:</th><td>" + mTestTypeName
                        + "<input type=\"hidden\" name=\"testTypeID\"
value=\""
                        + mTestTypeID + "\">"
```

```
                        + "<input type=\"hidden\" name=\"testTypeName\"
value=\""
                        + mTestTypeName + "\">\n</td></tr>\n");
            reply.append("<tr><th>Project:</th><td>" + mProjName
                        + "<input type=\"hidden\" name=\"projID\"
value=\"" + mProjID
                        + "\"></td></tr>\n");
            reply.append("</table>\n<p>\n<h4>Select a Parent Test
Folder</h4>\n");
            reply.append("<table border=\"1\"><tr><td>\n");
            reply.append(ServletUtil.getTestTreeFolders(mProjID));
            reply.append("<input type=\"hidden\" name=\"parentID\"
value=\"\">\n");
            reply.append("</td></tr></table><p>\n");
            reply.append(getTestTypeFormTable());
            reply.append("<input type=\"submit\"
value=\"Add\">\n</form>\n");
            showPage(req, resp, reply, getTestTypeJavaScript()
                        + ServletUtil.ADD_TEST_TREE_JS +
ServletUtil.ADD_TEST_TREE_CSS,
                        this);
      }

      /**
       * Helper method for getting appropriate
       * form table based upon test type selected
       * by user
       *
       * @return the String HTML form table
       */
      protected String getTestTypeFormTable() {
            String testTypeTable = "";
            if (mTestTypeName.equalsIgnoreCase("JUNIT")) {
                  testTypeTable = getJUnitFormTable();
            }
            return testTypeTable;
      }

      /**
       * Helper method for injecting the appropriate
       * JavaScript depending if a new Test folder or
       * definiton was selected by the user
       *
       * @return The JavaScript reference
       */
      protected String getTestTypeJavaScript() {
            String testTypeJS = "";
            if (mTestTypeName.equalsIgnoreCase("JUNIT")) {
                  testTypeJS = ServletUtil.ADD_TEST_TYPE_JS;
            } else if (mTestTypeName.equalsIgnoreCase("FOLDER")) {
                  testTypeJS = ServletUtil.ADD_TEST_FOLDER_JS;
            }
            return testTypeJS;
      }

      /**
       * Helper method gets the HTML for rendering
```

```
       * a JUnit test type input form table
       *
       * @return The HTML String form table
       */
      protected String getJUnitFormTable() {
            StringBuffer reply = new StringBuffer();
            reply
                        .append("<table >\n<caption><b>JUnit Test
Properties</b></caption>\n");
            reply
                        .append("<tr><th>Eclipse Home:</th><td><input
type=\"text\" name=\"eclipseHome\" size=\"75\" value=\""
                                    + ECLIPSE_HOME +
"\"></td></tr>\n");
            reply
                        .append("<tr><th>Eclipse
Workspace:</th><td><input type=\"text\" name=\"eclipseWorkspace\"
size=\"75\" value=\""
                                    + ECLIPSE_WORKSPACE +
"\"></td></tr>\n");
            reply
                        .append("<tr><th>Eclipse
Project:</th><td><input type=\"text\" name=\"eclipseProj\" size=\"75\"
value=\""
                                    + ECLIPSE_PROJECT +
"\"></td></tr>\n");
            reply
                        .append("<tr><th>Test Suite:</th><td><input
type=\"text\" name=\"testSuite\" size=\"75\" value=\""
                                    + mTestName +
".testsuite\"></td></tr>\n");
            reply
                        .append("<tr><th>Report
Directory:</th><td><input type=\"text\" name=\"reportDir\" size=\"75\"
value=\""
                                    + REPORT_DIR + "\"></td></tr>\n");
            reply
                        .append("<tr><th>TPTP Connection
URL:</th><td><input type=\"text\" name=\"tptpConn\" size=\"75\"
value=\""
                                    + TPTP_CONN + "\"></td></tr>\n");
            reply.append("</table>\n<br>\n");
            return reply.toString();
      }
}
```

AddTestEntity.java

```
/*********************************************************************
 *
 * File     :     AddTestEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet used for adding new Tests to the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import java.util.Hashtable;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestType;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tptp.TPTestCRUD;

/*********************************************************************
 * File           :     AddTestEntity.java
 *
 * Description    :     Servlet used for adding new Tests to the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class AddTestEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTestName = null;

      protected String mTestDesc = null;

      protected String mTestTypeID = null;

      protected String mTestTypeName = null;
```

```
	protected String mProjID = null;

	protected String mParentID = null;

	protected String mRemoteUser = null;

	protected Test mTestStub = null;

	public void init(ServletConfig config) throws ServletException {
		super.init(config);
	}

	/**
	 * Displays addition of new Test results,
	 * including errors to the user
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		mTestName = req.getParameter("testName");
		mTestDesc = req.getParameter("testDesc");
		mTestTypeID = req.getParameter("testTypeID");
		mTestTypeName = req.getParameter("testTypeName");
		mProjID = req.getParameter("projID");
		mParentID = req.getParameter("parentID");
		mRemoteUser = req.getRemoteUser();

		try {

			saveTest(req, resp);
			StringBuffer reply = new StringBuffer("<h3>Add Test "
+ mTestName
					+ " was Successful</h3>");
			showPage(req, resp, reply, null, this);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Creates a new Test based on input parameters
	 * and persists it to the database
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws Exception
	 */
```

```
	protected void saveTest(HttpServletRequest req,
HttpServletResponse resp)
			throws Exception {
		mTestStub = new Test();
		mTestStub.setName(mTestName);
		mTestStub.setDescription(mTestDesc);
		Test parent = new Test();
		parent.setId(Integer.parseInt(mParentID));
		parent.addChild(mTestStub);
		mTestStub.setParent(parent);
		TestType testType = new TestType();
		if (mTestTypeName.equalsIgnoreCase("Folder")) {
			testType.setName("Folder");
			mTestStub.setIsFolder('Y');
		} else if (mTestTypeName.equalsIgnoreCase("JUnit")) {
			mTestStub.setIsFolder('N');
			testType.setName("JUnit");
			JunitTest junit = new JunitTest();

	junit.setEclipseHome(req.getParameter("eclipseHome"));

	junit.setWorkspace(req.getParameter("eclipseWorkspace"));
			junit.setProject(req.getParameter("eclipseProj"));
			junit.setTestSuite(req.getParameter("testSuite"));
			junit.setReportDir(req.getParameter("reportDir"));

	junit.setTptpConnection(req.getParameter("tptpConn"));
			mTestStub.addJunitTest(junit);
		}
		mTestStub.setTestType(testType);

		TpteamUser addUser = new TpteamUser();
		addUser.setEcfId(Activator.getDefault().getTPBridgeClient()
				.getTPMgrECFID());
		mTestStub.setCreatedBy(addUser);
		Project proj = new Project();
		proj.setId(Integer.parseInt(mProjID));
		mTestStub.setProject(proj);
		String testXML = TestXML.getXML(mTestStub);

		// Wrap info into TPEvent
		Hashtable<String, String> dictionary = new
Hashtable<String, String>();
		dictionary.put(TPEvent.ID_KEY, String.valueOf(mParentID));
		dictionary.put(TPEvent.PROJECT_ID_KEY, mProjID);
		dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
				.getTPBridgeClient().getTPMgrECFID());
		dictionary.put(TPEvent.FROM, addUser.getEcfId());
		dictionary.put(TPEvent.TEST_XML_KEY, testXML);

		TPEvent tpEvent = new TPEvent(ITPBridge.TEST_ADD_REQ_TOPIC,
dictionary);

		TPTestCRUD.sendTestAddResponse(tpEvent);
	}
}
```

AddUser.java

```
/*********************************************************************
 *
 * File     :     AddUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for creating
 *                a new TPTeam User
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Role;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     AddUser.java
 *
 * Description    :     Servlet that displays an input form for
 *                      creating a new TPTeam User
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class AddUser extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;
      private boolean mIsProjAvailable = false;
      private boolean mIsRoleAvailable = false;
      private String mProjOptions = null;
      private String mRoleOptions = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gathers the Project and Role selection lists and
       * calls helper method to render data input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
```

```
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try
		{
		getProjOptions();
		getRoleOptions();
		if(mIsProjAvailable == false || mIsRoleAvailable == false)
		{
			throwError(req, resp);
		}
		else
		{
			showPage(req, resp);
		}
		}
		 catch (Exception e) {
				String error = "<h3>Error: " + e.getMessage() +
"<br>" +
				e.getCause() + "</h3>";
				adminError(req, resp, error);
				return;
			}
	}

	/**
	 * Helper method that gets all Projects in TPTeam
	 * and wraps into HTML select option tags
	 *
	 * @return The HTML Project option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	private String getProjOptions() throws Exception
	{
		Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
		Transaction tx = null;
		List<Project> projs = null;
		mProjOptions = "";
		try {

			tx = s.beginTransaction();

			projs = s.createQuery("from Project as p order by
p.name asc").list();
			for(Project proj : projs)
			{
				mProjOptions += "<option value=\"" +
proj.getId() + "\">" + proj.getName() + "</option>\n";
			}

			tx.commit();
		} catch (Exception e) {
```

```
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(!mProjOptions.equals(""))
                  mIsProjAvailable = true;

            mProjOptions = "<option selected>Choose Project</option>\n"
+  mProjOptions;
            return mProjOptions;
      }

      /**
       * Helper method that gets all Roles in TPTeam
       * and wraps into HTML select option tags
       *
       * @return The HTML Role option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getRoleOptions() throws Exception
      {
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<Role> roles = null;
            mRoleOptions = "";
            try {
                  tx = s.beginTransaction();

                  roles = s.createQuery("from Role as role order by
role.name asc").list();
                  for(Role role : roles)
                  {
                        mRoleOptions += "<option value=\"" +
role.getRoleId() + "\">" + role.getName() + "</option>\n";
                  }
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(!mRoleOptions.equals(""))
                  mIsRoleAvailable = true;

            mRoleOptions = "<option selected>Choose Role</option>\n" +
mRoleOptions;

            return mRoleOptions;
      }

      /**
       * Helper method that renders HTML error messages
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
```

```
	 * @throws ServletException
	 * @throws IOException
	 */
	private void throwError(HttpServletRequest req,
HttpServletResponse resp)
	throws ServletException, IOException
	{
		String error = "<h3>Error: No Project or TPTeam Role
Available</h3>";
		adminError(req, resp, error);
	}

	/**
	 * Helper method that renders the HTML form
	 * as a table
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws ServletException
	 * @throws IOException
	 */
	private void showPage(HttpServletRequest req, HttpServletResponse
resp)
	throws ServletException, IOException
	{
		String reply = "<h4>Add User</h4>\n";
		reply += "<form name=\"addUser\" method=\"post\"
onSubmit=\"return validateForm( );\">\n";
		reply += "<table ><tr><th>First Name:</th><td><input
type=\"text\" name=\"firstName\" size=\"25\"></td></tr>\n";
		reply += "<tr><th>Last Name:</th><td><input type=\"text\"
name=\"lastName\" size=\"25\"></td></tr>\n";
		reply += "<tr><th>User Name:</th><td><input type=\"text\"
name=\"userName\" size=\"25\"></td></tr>\n";
		reply += "<tr><th>Password:</th><td><input
type=\"password\" name=\"password\" size=\"25\"></td></tr>\n";
		reply += "<tr><th>Email:</th><td><input type=\"text\"
name=\"email\" size=\"50\"></td></tr>\n";
		reply += "<tr><th>Phone (ddd-ddd-dddd):</th><td><input
type=\"text\" name=\"phone\" size=\"25\"></td></tr>\n";
		reply += "<tr><th>ECF ID:</th><td><input type=\"text\"
name=\"ecfID\" size=\"50\"></td></tr>\n";
		reply += "<tr><th>Role:</th><td><select name=\"role\">" +
mRoleOptions + "</select></td></tr>\n";
		reply += "<tr><th>Projects:</th><td><select multiple
size=\"5\" name=\"projects\">" + mProjOptions +
"</select></td></tr>\n";
		reply += "</table>\n<br>\n<input type=\"submit\"
value=\"Add\">\n</form>\n";

		adminHeader(req, resp, ADD_USER_JS);
		adminReply(req, resp, reply);
		adminFooter(req, resp);
	}


}
```

AddUserEntity.java

```
/**********************************************************************
 *
 * File     :     AddUserEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that creates a new User and persists
 *                      it to the TPTeam database
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add;

import java.io.IOException;
import java.util.Date;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Role;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     AddUserEntity.java
 *
 * Description    :     Servlet that creates a new User and persists
 *                      it to the TPTeam database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class AddUserEntity extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;
      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Creates an new User from input parameters, persists
       * it to the database.  Renders results, including errors,
       * to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
```

```
                  throws ServletException, IOException {
            String firstName = req.getParameter("firstName");
            String lastName = req.getParameter("lastName");
            String userName = req.getParameter("userName");
            String password = req.getParameter("password");
            String email = req.getParameter("email");
            String ecfId = req.getParameter("ecfID");
            String phone = req.getParameter("phone");
            String roleId = req.getParameter("role");
            String[] projIds = req.getParameterValues("projects");
            Transaction tx = null;
            try {
                  int remoteUserId =
ServletUtil.getRemoteUserID(req.getRemoteUser());
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = new TpteamUser();
                  user.setFirstName(firstName);
                  user.setLastName(lastName);
                  user.setUserName(userName);
                  user.setPassword(ServletUtil.getSHA1Hash(password));
                  user.setEmail(email);
                  user.setPhone(phone);
                  user.setEcfId(ecfId);
                  user.setCreatedBy(remoteUserId);
                  user.setCreatedDate(new Date());
                  Role role = (Role) s.load(Role.class, new
Integer(roleId));
                  user.setRole(role);
                  for (String projId : projIds) {
                        Project project = (Project)
s.load(Project.class, new Integer(
                                    projId));
                        project.addToTpteamUsers(user);
                        user.getProjects().add(project);
                  }
                  s.save(user);
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
            adminHeader(req, resp, null);
            String reply = "<h3>Add TPTeam User " + userName
                        + " was Successful</h3>";
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }

}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete

DeleteProduct.java

```
/**********************************************************************
 *
 * File     :     DeleteProduct.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Product
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     DeleteProduct.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam Product
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

**********************************************************************/
public class DeleteProduct extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;
      private boolean mIsProdAvailable = false;
      private String mProdRows = null;

      private String rowNameHeader = "<tr><form method=\"post\"
onSubmit=\"return validateForm(this);\"><th>Name:</th><td><input
type=\"hidden\" name=\"prodName\" size=\"25\"";

      private String rowDescHeader = "<th>Description:</th><td>";

      private String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Delete\"></td>\n</form></tr>\n";
```

```
      private String rowIDHeader = "<input type=\"hidden\"
name=\"prodId\"";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gathers the Product selection list, renders input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try
            {
            getProdRows();
            if(mIsProdAvailable == false)
            {
                  throwError(req, resp);
            }
            else
            {
                  showPage(req, resp);
            }
            }
             catch (Exception e) {
                        String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                        e.getCause() + "</h3>";
                        adminError(req, resp, error);
                        return;
                  }
      }

      /**
       * Gets all TPTeam Products and wraps them in HTML
       * selection list option tags
       *
       * @return the HTML String Product option tag
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getProdRows() throws Exception
      {
            Session s =
HibernateUtil.getSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<Product> prods = null;
            StringBuffer prodRows = new StringBuffer();
            try {

                  tx = s.beginTransaction();
```

```
                  prods = s.createQuery("from Product as p order by
p.name asc").list();
                  for(Product prod : prods)
                  {
                        String desc = prod.getDescription();
                        if(desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        prodRows.append(rowNameHeader + " value=\"" +
prod.getName() + "\">" + prod.getName() + "</td>\n");
                        prodRows.append(rowDescHeader +
prod.getDescription() + "</td>\n");
                        prodRows.append(rowIDHeader + " value=\"" +
prod.getId() + "\">\n");
                        prodRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(prodRows.length() > 0)
                  mIsProdAvailable = true;
            mProdRows = prodRows.toString();
            return mProdRows;
      }

      /**
       * Helper method that renders errors as HTML
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
      throws ServletException, IOException
      {
            String error = "<h3>Error: No Product Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method that renders HTML input form
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
      throws ServletException, IOException
      {
```

```
            String reply = "<h4>Delete Product</h4>\n<table>" +
mProdRows + "</table>";
            adminHeader(req, resp, DELETE_PROD_JS);
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

DeleteProductEntity.java

```
/*********************************************************************
 *
 * File     :     DeleteProductEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that deletes a Product from the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.hibernate.Session;
import org.hibernate.Transaction;

import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     DeleteProductEntity.java
 *
 * Description    :     Servlet that deletes a Product from the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class DeleteProductEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Deletes a Product based on ID from selection list.
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {

            String  name = req.getParameter("prodName");
            String prodId = req.getParameter("prodId");

            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {
                  tx = s.beginTransaction();
                  Product prod = (Product) s.load(Product.class, new
Integer(prodId));
                  s.delete(prod);
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                  e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
                  adminHeader(req, resp, null);
                  String reply = "<h3>Delete Product " + name + " was
Successful</h3>";
                  adminReply(req, resp, reply);
                  adminFooter(req, resp);
      }


}
```

DeleteProject.java

```
/**********************************************************************
 *
 * File     :     DeleteProject.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Project
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     DeleteProject.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class DeleteProject extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;
      private boolean mIsProjAvailable = false;
      private String mProjRows = null;

      private String rowNameHeader = "<tr><form method=\"post\"
onSubmit=\"return validateForm(this);\"><th>Project</th><td>";

      private String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      private String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Delete\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gathers all Projects as HTML input forms, renders page
```

```
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try
		{
		getProjRows();
		if(mIsProjAvailable == false)
		{
			throwError(req, resp);
		}
		else
		{
			showPage(req, resp);
		}
		}
		 catch (Exception e) {
				String error = "<h3>Error: " + e.getMessage() +
"<br>" +
				e.getCause() + "</h3>";
				adminError(req, resp, error);
				return;
			}
	}

	/**
	 * Gets all Projects and wraps them as HTML
	 * selection list options
	 *
	 * @return the HTML String of Project options
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	private String getProjRows() throws Exception
	{
		Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
		Transaction tx = null;
		List<Project> projs = null;
		StringBuffer projRows = new StringBuffer();
		try {

			tx = s.beginTransaction();

			projs = s.createQuery("from Project as p order by
p.name asc").list();
			for(Project proj : projs)
			{
				projRows.append(rowNameHeader + proj.getName()
+ "<input type=\"hidden\" name=\"projName\" value=\"" + proj.getName()
+ "\">");
```

```
                        if(proj.getDescription() != null &&
!proj.getDescription().equalsIgnoreCase("null"))
                        {
                              projRows.append(": " +
proj.getDescription());
                        }
                        projRows.append("</td>\n" + rowIDHeader + "
value=\"" + proj.getId() + "\">\n" + rowSubmitHeader + "\n");
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(projRows.length() > 0)
                  mIsProjAvailable = true;

            mProjRows = projRows.toString();

            return mProjRows;
      }

      /**
       * Helper method to render errors as HTML
       * @param req The Servlet Request
       * @param resp The Servlet Response
       *
       * @throws ServletException
       * @throws IOException
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
      throws ServletException, IOException
      {
            String error = "<h3>Error: No Project Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method to render the HTML page, including
       * JavaScript
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
      throws ServletException, IOException
      {
            String reply = "<h4>Delete Project</h4>\n<table
border=\"2\">" + mProjRows + "</table>";
            adminHeader(req, resp, DELETE_PROJ_JS);
            adminReply(req, resp, reply);
```

```
            adminFooter(req, resp);
        }
    }
```

DeleteProjectEntity.java

```
/*********************************************************************
 *
 * File     :     DeleteProjectEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that deletes a Project from the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     DeleteProjectEntity.java
 *
 * Description    :     Servlet that deletes a Project from the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class DeleteProjectEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Deletes a Project based on ID from selection list.
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            String projId = req.getParameter("projId");
```

```
            String name = req.getParameter("projName");
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {

                  tx = s.beginTransaction();
                  Project proj = (Project)s.load(Project.class, new
Integer(projId));

      proj.getTpteamUsers().removeAll(proj.getTpteamUsers());
                  s.delete(proj);
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
            adminHeader(req, resp, null);
            String reply = "<h3>Delete Project " + name + " was
Successful</h3>";
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

DeleteTest.java

```
/**********************************************************************
 *
 * File     :     DeleteTest.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Test by first selecting its parent Project
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     DeleteTest.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam Test by first selecting its
 *                      parent Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class DeleteTest extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProjAvailable = false;

      protected String mProjRows = null;

      protected String mRemoteUser = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"deleteTest2\"><th>Project</th><td>";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Delete Test Node\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
```

```
            super.init(config);
      }

      /**
       * Gathers Project select list form and renders so user
       * can select parent Project of Test to be deleted
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mRemoteUser = req.getRemoteUser();
                  getProjRows();
                  if (mIsProjAvailable == false) {
                        StringBuffer error = new StringBuffer(
                                    "<h3>Error: No Project
Available</h3>");
                        throwError(req, resp, error, this);
                  } else {
                        StringBuffer reply = new StringBuffer(
                                    "<h4>Delete Test Tree Node: Select
Parent Project</h4>\n<table border=\"2\">"
                                                + mProjRows +
"</table>");
                        showPage(req, resp, reply, null, this);
                  }
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Gets all TPTeam Projects and wraps them
       * into HTML select option tags
       *
       * @return The Project select option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      protected String getProjRows() throws Exception {
            Session s = Activator.getDefault().getHiberSessionFactory()
                        .getCurrentSession();
            Transaction tx = null;
            List<Project> projs = null;
            StringBuffer projRows = new StringBuffer();
            try {

                  tx = s.beginTransaction();
```

```
                  projs = s.createQuery("from Project as p order by
p.name asc")
                              .list();
                  for (Project proj : projs) {
                        String desc = proj.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId()
                                    + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;
            mProjRows = projRows.toString();
            return mProjRows;
      }
}
```

DeleteTest2.java

```
/*********************************************************************
 *
 * File     :     DeleteTest2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Test by selecting it from the Test tree
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     DeleteTest2.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam Test by selecting it from the
 *                      Test tree
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class DeleteTest2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      private String mProjID = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gets the ID of the Project selected and renders test tree for
       * Test selection
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
```

```
                mProjID = req.getParameter("projId");
                getPage(req, resp);
            } catch (Exception e) {
                StringBuffer error = new StringBuffer("<h3>Error: "
                            + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                throwError(req, resp, error, this);
                return;
            }
      }

      /**
       * Helper method for rendering a Project's Test tree
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      private void getPage(HttpServletRequest req, HttpServletResponse
resp)
                throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply
                        .append("<h4>Select a Folder or Test, then
Click the Delete Button</h4>\n");
            reply
                        .append("<form name=\"deleteTest\"
method=\"post\" onSubmit=\"return validateForm(this);\">\n");
            reply.append("<table border=\"1\"><tr><td>\n");
            reply.append(ServletUtil.getTestTree(mProjID, false));
            reply
                        .append("<input type=\"hidden\" name=\"testID\"
value=\"\">\n</td></tr></table><p>\n<input type=\"submit\"
value=\"Delete\">\n</form>\n");

            showPage(req, resp, reply, ServletUtil.DELETE_TEST_TREE_JS
                        + ServletUtil.ADD_TEST_TREE_JS +
ServletUtil.ADD_TEST_TREE_CSS,
                        this);
      }
}
```

DeleteTestEntity.java

```
/*********************************************************************
 *
 * File     :     DeleteTestEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that deletes a Test from the TPTeam
 *                database
 *
 ********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import java.util.Hashtable;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tptp.TPTestCRUD;

/*********************************************************************
 * File           :     DeleteTestEntity.java
 *
 * Description    :     Servlet that deletes a Test from the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 ********************************************************************/
public class DeleteTestEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTestID = null;

      protected String mTestName = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Collects the ID of the Test selected, requests
       * Test deletion, renders results to user
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
```

```
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            mTestID = req.getParameter("testID");

            try {
                  deleteTest();
                  StringBuffer reply = new StringBuffer("<h3>Delete
Test Node \""
                              + mTestName + " \" was Successful</h3>");
                  showPage(req, resp, reply, null, this);
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Helper method, creates a test deletion
       * request TPEvent and calls upon TPTestCRUD to
       * delete the Test
       *
       * @see TPTestCRUD
       * @throws Exception
       */
      protected void deleteTest() throws Exception {
            // Wrap info into TPEvent and send delete request
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY, String.valueOf(mTestID));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient().getTPMgrECFID());
            TPEvent tpEvent = new TPEvent(ITPBridge.TEST_DEL_REQ_TOPIC,
dictionary);
            TPTestCRUD.deleteTest(tpEvent);
            mTestName =
tpEvent.getDictionary().get(TPEvent.TEST_NAME_KEY);
            TPTestCRUD.sendDelTestResponse(tpEvent);
      }
}
```

DeleteUser.java

```
/*********************************************************************
 *
 * File     :     DeleteUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                      a TPTeam User
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     DeleteUser.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam User
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class DeleteUser extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      private boolean mIsUserAvailable = false;

      private String mUserRows = null;

      private String rowNameHeader = "<tr><form method=\"post\"
onSubmit=\"return validateForm(this);\"><td>";

      private String rowIDHeader = "<input type=\"hidden\"
name=\"userId\"";

      private String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Delete\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
	/**
	 * Gathers the User selection list, renders input form
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mUserRows = getUserRows();
			if (mIsUserAvailable == false) {
				throwError(req, resp);
			} else {
				showPage(req, resp);
			}
		} catch (Exception e) {
			String error = "<h3>Error: " + e.getMessage() +
"<br>"
					+ e.getCause() + "</h3>";
			adminError(req, resp, error);
			return;
		}
	}

	/**
	 * Gets all TPTeam Users and wraps them in HTML
	 * selection list option tags
	 *
	 * @return the HTML String User option tag
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	private String getUserRows() throws Exception {
		Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
		Transaction tx = null;
		List<TpteamUser> users = null;
		StringBuffer userRows = new StringBuffer();
		try {

			tx = s.beginTransaction();

			users = s.createQuery(
					"from TpteamUser as user order by
user.lastName asc")
					.list();
			for (TpteamUser user : users) {
				userRows.append(rowNameHeader +
user.getLastName() + ", "
						+ user.getFirstName() + "</td>\n");
				userRows.append(rowIDHeader + " value=\"" +
user.getId()
						+ "\">\n");
				userRows.append(rowSubmitHeader);
```

```
                }

                tx.commit();
            } catch (Exception e) {
                if (tx != null)
                    tx.rollback();
                throw e;
            }
            if (userRows.length() > 0)
                mIsUserAvailable = true;

            return userRows.toString();
        }

        /**
         * Helper method that renders errors as HTML
         * @param req The Servlet Request
         * @param resp The Servlet Response
         * @throws ServletException
         * @throws IOException
         */
        private void throwError(HttpServletRequest req,
HttpServletResponse resp)
                throws ServletException, IOException {
            String error = "<h3>Error: No Users Exist</h3>";
            adminError(req, resp, error);
        }

        /**
         * Helper method that renders HTML input form
         * @param req The Servlet Request
         * @param resp The Servlet Response
         * @throws ServletException
         * @throws IOException
         */
        private void showPage(HttpServletRequest req, HttpServletResponse
resp)
                throws ServletException, IOException {
            String reply = "<h4>Delete User</h4>\n<table
border=\"2\">\n<th>Last, First</th><th></th>\n"
                    + mUserRows + "</table>";
            adminHeader(req, resp, DELETE_USER_JS);
            adminReply(req, resp, reply);
            adminFooter(req, resp);
        }
}
```

DeleteUserEntity.java

```
/*********************************************************************
 *
 * File     :     DeleteUserEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that deletes a User from the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     DeleteUserEntity.java
 *
 * Description    :     Servlet that deletes a User from the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class DeleteUserEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      private String mLastName = null;

      private String mFirstName = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Deletes a User based on ID from selection list.
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
```

```
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		String userId = req.getParameter("userId");
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		Transaction tx = null;
		try {
			tx = s.beginTransaction();
			TpteamUser user =
(TpteamUser)s.load(TpteamUser.class, new Integer(userId));
			mLastName = user.getLastName();
			mFirstName = user.getFirstName();
			for(Project proj : user.getProjects())
			{
				proj.getTpteamUsers().remove(user);

			}
			s.delete(user);
			s.flush();
			tx.commit();
		} catch (Exception e) {
			if (tx != null)
				tx.rollback();
			String error = "<h3>Error: " + e.getMessage() +
"<br>"
					+ e.getCause() + "</h3>";
			adminError(req, resp, error);
			return;
		}
		adminHeader(req, resp, null);
		String reply = "<h3>Deletion of TPTeam User " + mLastName +
", " + mFirstName
				+ " was Successful</h3>";
		adminReply(req, resp, reply);
		adminFooter(req, resp);
	}
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec

AdminProcessTestExec.java

```
/*********************************************************************
 *
 * File     :     AdminProcessTestExec.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that validates a test request
 *                and delegates its execution
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec;

import java.io.IOException;
import java.util.Hashtable;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tptp.TPTestExec;

/*********************************************************************
 * File           :     AdminProcessTestExec.java
 *
 * Description    :     Servlet that validates a test request
 *                      and delegates its execution
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class AdminProcessTestExec extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mRemoteUser;

      protected String mTestID;

      protected TPEvent mTPEvent;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
```

```
	}

	/**
	 * Collects ID of Test be executed, delegates execution,
	 * and renders execution results
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {

		mRemoteUser = req.getRemoteUser();
		mTestID = req.getParameter("testID");

		try {
			validateTestReq(req);
			StringBuffer reply = new StringBuffer(execTest());
			showPage(req, resp, reply, null, this);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Creates a new test execution request TPEvent and requests
	 * and execution from the TPTestExec class
	 *
	 * @see TPTestExec
	 * @return the test result in HTML format
	 * @throws Exception
	 */
	protected String execTest() throws Exception {

		Hashtable<String, String> dictionary = new
Hashtable<String, String>();
		String fromECFID =
Activator.getDefault().getTPBridgeClient()
				.getTPMgrECFID();
		dictionary.put(TPEvent.FROM, fromECFID);
		mTPEvent = new TPEvent(ITPBridge.TEST_EXEC_REQ_TOPIC,
dictionary);

		TPTestExec.runTest(mTestID, mTPEvent);
		TPTestExec.sendTestExecResponse(mTPEvent);

		StringBuffer reply = new StringBuffer();
		reply
				.append("<h3 align=\"center\">TPManager Test
Execution Result Page</h3>");
```

```
            reply.append("<h3>Test Name: " + mTPEvent.getTestName() +
"</h3>");
            reply.append("<h3>Exec Username: " + mRemoteUser
                        + "</h3>\n<h3>testID: " + mTestID + "</h3>");
            reply.append("<h3>TestSuite Execution Verdict: "
                        +
mTPEvent.getDictionary().get(TPEvent.VERDICT_KEY) + "</h3>");

            return reply.toString();
      }

      /**
       * Helper method that ensures a single test definition and not
       * a folder was selected for execution
       *
       * @param req The Servlet Request
       * @throws Exception
       */
      protected void validateTestReq(HttpServletRequest req) throws
Exception {
            Transaction tx = null;
            try {
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  Test test = (Test) s.load(Test.class, new
Integer(mTestID));

                  if (Integer.parseInt(mTestID) <= 0 ||
test.getIsFolder() == 'Y'
                              || test.getIsFolder() == 'y') {
                        throw new Exception(
                                    "Invalid Test Definition Selected
for Execution",
                                    new Throwable(
                                                "A Test Folders was
Selected for Execution"));
                  }

                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
      }
}
```

ExecTest.java

```
/*********************************************************************
 *
 * File     :     ExecTest.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for executing
 *                a TPTeam Test by first selecting its parent Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec;

import java.io.IOException;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;

/*********************************************************************
 * File           :     ExecTest.java
 *
 * Description    :     Servlet that displays an input form for
 *                      executing a TPTeam Test by first selecting its
 *                      parent Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ExecTest extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;
      protected boolean mIsProjAvailable = false;
      protected String mProjRows = null;
      protected String mRemoteUser = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"execTest2\"><th align=\"left\">Project</th><td
align=\"right\">";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"View Test Tree\"></td>\n</form></tr>\n";
```

```
	public void init(ServletConfig config) throws ServletException {
		super.init(config);
	}

	/**
	 * Gathers Project select list form and renders so user
	 * can select parent Project of Test to be executed
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try
		{
		mRemoteUser = req.getRemoteUser();
		getProjRows();
		if(mIsProjAvailable == false)
		{
			StringBuffer error = new StringBuffer("<h3>Error: No
Project Available</h3>");
			throwError(req, resp, error, this);
		}
		else
		{
			StringBuffer reply = new StringBuffer("<h4>Execute
Test: Select Parent Project</h4>\n<table border=\"2\">" + mProjRows +
"</table>");
			showPage(req, resp, reply, null, this);
		}
		}
		 catch (Exception e) {
				String error = "<h3>Error: " + e.getMessage() +
"<br>" +
				e.getCause() + "</h3>";
				adminError(req, resp, error);
				return;
			}
	}

	/**
	 * Gets all TPTeam Projects and wraps them
	 * into HTML select option tags
	 *
	 * @return The Project select option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getProjRows() throws Exception
	{
		Transaction tx = null;
		Set<Project> projs = new HashSet<Project>();
		StringBuffer projRows = new StringBuffer();
```

```
            int remoteUserId = -1;
            try {
                  if (this instanceof UserServlet) {
                        remoteUserId = getRemoteUserID(mRemoteUser);
                  }
                  Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
                  tx = s.beginTransaction();
                  if (remoteUserId == -1) {
                        List<Project> projList = s.createQuery(
                                    "from Project as p order by p.name
asc").list();
                        projs.addAll(projList);
                  } else {
                        TpteamUser user = (TpteamUser) s.createQuery(
                                    "from TpteamUser as user where
user.id = "
                                                +
remoteUserId).uniqueResult();
                        projs = user.getProjects();
                  }
                  for(Project proj : projs)
                  {
                        String desc = proj.getDescription();
                        if(desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId() + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(projRows.length() > 0)
                  mIsProjAvailable = true;
            mProjRows = projRows.toString();
            return mProjRows;
      }
}
```

ExecTest2.java

```
/*********************************************************************
 *
 * File     :     ExecTest2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for executing
 *                a TPTeam Test by selecting it from the Test tree
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     ExecTest2.java
 *
 * Description    :     Servlet that displays an input form for
 *                      executing a TPTeam Test by selecting it from
 *                      the Test tree
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ExecTest2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mProjID = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gets the ID of the Project selected and renders test tree for
       * Test selection
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mProjID = req.getParameter("projId");
```

```
                getPage(req, resp);
            } catch (Exception e) {
                StringBuffer error = new StringBuffer("<h3>Error: " +
e.getMessage() + "<br>"
                        + e.getCause() + "</h3>");
                throwError(req, resp, error, this);
                return;
            }
      }

      /**
       * Helper method for rendering a Project's Test tree
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      protected void getPage(HttpServletRequest req,
HttpServletResponse resp)
                throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply
                        .append("<h4>Select a Test from the Project
Tree Below, then Click the Execute Button</h4>\n");
            reply
                        .append("<form name=\"execTest\"
method=\"post\" onSubmit=\"return validateForm(this);\">\n");

            reply.append("<table border=\"1\"><tr><td>\n");

            reply.append(ServletUtil.getTestTree(mProjID, false));

            reply
                        .append("<input type=\"hidden\" name=\"testID\"
value=\"\">\n</td></tr></table><p>\n<input type=\"submit\"
value=\"Execute\">\n</form>\n");

            showPage(req, resp, reply, ServletUtil.EXEC_TEST_TREE_JS
                        + ServletUtil.ADD_TEST_TREE_JS +
ServletUtil.ADD_TEST_TREE_CSS, this);
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update

UpdateProduct.java

```
/**********************************************************************
 *
 * File     :     UpdateProduct.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Product
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     UpdateProduct.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam Product
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class UpdateProduct extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      private boolean mIsProdAvailable = false;

      private String mProdRows = null;

      private String rowNameHeader = "<tr><form method=\"post\"
onSubmit=\"return validateForm(this);\"><th>Name:</th><td><input
type=\"text\" name=\"prodName\" size=\"25\"";

      private String rowDescHeader = "<th>Description:</th><td><input
type=\"text\" name=\"prodDesc\" size=\"50\"";
```

```
      private String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Update\"></td>\n</form></tr>\n";

      private String rowIDHeader = "<input type=\"hidden\"
name=\"prodId\"";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Renders the new Product input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  getProdRows();
                  if (mIsProdAvailable == false) {
                        throwError(req, resp);
                  } else {
                        showPage(req, resp);
                  }
            } catch (Exception e) {
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
      }

      /**
       * Helper method that gets all Products and wraps
       * them in HTML option tags
       *
       * @return The Product option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getProdRows() throws Exception {
            Session s = Activator.getDefault().getHiberSessionFactory()
                        .getCurrentSession();
            // For standalone
            // Session s =
HibernateUtil.getSessionFactory().getCurrentSession();

            Transaction tx = null;
            List<Product> prods = null;
            StringBuffer prodRows = new StringBuffer();
            try {

                  tx = s.beginTransaction();
```

```
                  prods = s.createQuery("from Product as p order by
p.name asc")
                              .list();
                  for (Product prod : prods) {
                        String desc = prod.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        prodRows.append(rowNameHeader + " value=\"" +
prod.getName()
                                    + "\"></td>\n");
                        prodRows.append(rowDescHeader + " value=\"" +
desc
                                    + "\"></td>\n");
                        prodRows.append(rowIDHeader + " value=\"" +
prod.getId()
                                    + "\">\n");
                        prodRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (prodRows.length() > 0)
                  mIsProdAvailable = true;
            mProdRows = prodRows.toString();
            return mProdRows;
      }

      /**
       * Helper method to render errors as HTML
       * @param req The Servlet Request
       * @param resp The Servlet Response
       *
       * @throws ServletException
       * @throws IOException
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException, IOException {
            String error = "<h3>Error: No Product Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method to render the HTML page, including
       * JavaScript
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
```

```
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            String reply = "<h4>Update Product</h4>\n<table>" +
mProdRows
                        + "</table>";
            adminHeader(req, resp, UPDATE_PROD_JS);
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

UpdateProductEntity.java

```
/*********************************************************************
 *
 * File     :     UpdateProductEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a Product in the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateProductEntity.java
 *
 * Description    :     Servlet that updates a Product in the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateProductEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Updates a Product with the passed in form inputs
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
```

```
            String  name = req.getParameter("prodName");
            String description = req.getParameter("prodDesc");
            String prodId = req.getParameter("prodId");

            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {
                  tx = s.beginTransaction();
                  Product prod = (Product) s.load(Product.class, new
Integer(prodId));
                  prod.setName(name);
                  prod.setDescription(description);
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                  e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
                  adminHeader(req, resp, null);
                  String reply = "<h3>Update Product " + name + " was
Successful</h3>";
                  adminReply(req, resp, reply);
                  adminFooter(req, resp);
      }
}
```

UpdateProject.java

```
/*********************************************************************
 *
 * File     :     UpdateProject.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateProject.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateProject extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      private boolean mIsProjAvailable = false;

      private String mProjRows = null;

      private String rowNameHeader = "<tr><form method=\"post\"
action=\"updateProject2\"><th>Project</th><td>";

      private String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      private String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Update\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
	/**
	 * Renders the update Project input form for selecting one
Project
	 *
	 * @param req
	 *            The Servlet Request
	 * @param resp
	 *            The Servlet Response
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			getProjRows();
			if (mIsProjAvailable == false) {
				throwError(req, resp);
			} else {
				showPage(req, resp);
			}
		} catch (Exception e) {
			String error = "<h3>Error: " + e.getMessage() +
"<br>"
					+ e.getCause() + "</h3>";
			adminError(req, resp, error);
			return;
		}
	}

	/**
	 * Helper method that gets all Project and wraps them in HTML
option tags
	 *
	 * @return The Product option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	private String getProjRows() throws Exception {
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		// For standalone
		// Session s =
HibernateUtil.getSessionFactory().getCurrentSession();

		Transaction tx = null;
		List<Project> projs = null;
		StringBuffer projRows = new StringBuffer();
		try {

			tx = s.beginTransaction();

			projs = s.createQuery("from Project as p order by
p.name asc")
					.list();
			for (Project proj : projs) {
				projRows.append(rowNameHeader +
proj.getName());
				if (proj.getDescription() != null
```

```
                                    &&
!proj.getDescription().equalsIgnoreCase("null")) {
                              projRows.append(": " +
proj.getDescription());
                        }
                        projRows.append("</td>\n" + rowIDHeader + "
value=\""
                                    + proj.getId() + "\">\n" +
rowSubmitHeader + "\n");
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;

            mProjRows = projRows.toString();

            return mProjRows;
      }

      /**
       * Helper method to render errors as HTML
       *
       * @param req
       *            The Servlet Request
       * @param resp
       *            The Servlet Response
       *
       * @throws ServletException
       * @throws IOException
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException, IOException {
            String error = "<h3>Error: No Project Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method to render the HTML page, including JavaScript
       *
       * @param req
       *            The Servlet Request
       * @param resp
       *            The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
```

```
            String reply = "<h4>Update Project</h4>\n<table
border=\"2\">"
                        + mProjRows + "</table>";
            adminHeader(req, resp, null);
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

UpdateProject2.java

```
/*********************************************************************
 *
 * File     :     UpdateProject2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the details for updating
 *                a particular TPTeam Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateProject2.java
 *
 * Description    :     Servlet that displays the details for updating
 *                      a particular TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateProject2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;
      private boolean mIsProdAvailable = false;
      private boolean mIsTeamUserAvailable = false;
      private String mProdOptions = null;
      private String mTeamOptions = null;
      private String mProjId = null;
      private String mProjName = null;
      private String mProjDesc = null;
      private int mProdId = 0;
      private int[] mUserIds = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
```

```
       * Gathers the Project selected, the Product, and User
       * selection lists, renders input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try
            {
            mProjId = req.getParameter("projId");
            getProject();
            getProdOptions();
            getTeamOptions();
            if(mIsProdAvailable == false || mIsTeamUserAvailable ==
false)
            {
                  throwError(req, resp);
            }
            else
            {
                  showPage(req, resp);
            }
            }
             catch (Exception e) {
                        String error = "<h3>Error: " + e.getMessage() +
"<br>" +
                        e.getCause() + "</h3>";
                        adminError(req, resp, error);
                        return;
                  }
      }

      /**
       * Loads a Project from the database, populates
       * member variables for TPTeam user and Product IDs
       *
       * @throws Exception
       */
      private void getProject() throws Exception
      {
            Project proj = null;
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {

                  tx = s.beginTransaction();
                  proj = (Project)s.load(Project.class, new
Integer(mProjId));
                  if(proj.getDescription() != null)
                  {
                        mProjDesc = proj.getDescription();
                  }
```

```
                  else
                  {
                        mProjDesc = "";
                  }
                  mProjName = proj.getName();
                  mProdId = proj.getProduct().getId();
                  mUserIds = new int[proj.getTpteamUsers().size()];
                  int idx = 0;
                  for(TpteamUser user : proj.getTpteamUsers())
                  {
                        mUserIds[idx++] = user.getId();
                  }
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
      }

      /**
       * Helper method that gets all Products and wraps
       * them in HTML option tags
       *
       * @return The Product option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getProdOptions() throws Exception
      {
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<Product> prods = null;
            StringBuffer prodOptions = new StringBuffer();

            try {

                  tx = s.beginTransaction();

                  prods = s.createQuery("from Product as p order by
p.name asc").list();
                  for(Product prod : prods)
                  {
                        if(prod.getId() == mProdId)
                        {
                              prodOptions.append("<option selected
value=\"" + prod.getId() + "\">" + prod.getName() + "</option>");
                        }
                        else
                        {
                              prodOptions.append("<option value=\"" +
prod.getId() + "\">" + prod.getName() + "</option>");
                        }
                  }
```

```
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(prodOptions.length() > 0)
                  mIsProdAvailable = true;

            mProdOptions = prodOptions.toString();

            return mProdOptions;
      }

      /**
       * Helper method that gets all TPTeam users and
       * wraps them into HTML option tags
       *
       * @return The TPTeam user option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      private String getTeamOptions() throws Exception
      {
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            Transaction tx = null;
            List<TpteamUser> users = null;
            StringBuffer teamOptions = new StringBuffer();
            try {

                  tx = s.beginTransaction();

                  users = s.createQuery("from TpteamUser as team order
by team.lastName asc").list();
                  for(TpteamUser user : users)
                  {
                        if(isUserSelected(user))
                        {
                              teamOptions.append("<option selected
value=\"" + user.getId() + "\">" + user.getLastName() + ", " +
user.getFirstName()+ "</option>");
                        }
                        else
                        {
                              teamOptions.append("<option value=\"" +
user.getId() + "\">" + user.getLastName() + ", " + user.getFirstName()+
"</option>");
                        }
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
```

```
            if(teamOptions.length()> 0)
                  mIsTeamUserAvailable = true;

            mTeamOptions = teamOptions.toString();

            return mTeamOptions;
      }

      private boolean isUserSelected(TpteamUser user)
      {
            boolean returnVal = false;
            for(int selectedUserId : mUserIds)
            {
                  if(user.getId() == selectedUserId)
                  {
                        returnVal = true;
                        break;
                  }
            }
            return returnVal;
      }

      /**
       * Helper method to render errors as HTML
       * @param req The Servlet Request
       * @param resp The Servlet Response
       *
       * @throws ServletException
       * @throws IOException
       */
      private void throwError(HttpServletRequest req,
HttpServletResponse resp)
      throws ServletException, IOException
      {
            String error = "<h3>Error: No Product or TPTeam Member
Available</h3>";
            adminError(req, resp, error);
      }

      /**
       * Helper method to render the HTML page, including
       * JavaScript
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      private void showPage(HttpServletRequest req, HttpServletResponse
resp)
      throws ServletException, IOException
      {
            StringBuffer reply = new StringBuffer();
            reply.append("<h4>Update Project</h4>\n");
            reply.append("<form name=\"updateProj\" method=\"post\"
onSubmit=\"return validateForm(this);\">\n");
```

```
            reply.append("<table ><tr><th>Name:</th><td><input
type=\"text\" name=\"projName\"  value=\"" + mProjName + "\"
size=\"25\"></td></tr>\n");
            reply.append("<tr><th>Description:</th><td><input
type=\"text\" name=\"projDesc\" value=\"" + mProjDesc + "\"
size=\"50\"></td></tr>\n");
            reply.append("<tr><th>Product:</th><td><select
name=\"prod\">" + mProdOptions + "</select></td></tr>\n");
            reply.append("<tr><th>Team Members:</th><td><select
multiple size=\"5\" name=\"team\">" + mTeamOptions +
"</select></td></tr>\n");
            reply.append("</table>\n<br>\n<input type=\"hidden\"
name=\"projId\" value=\"" + mProjId + "\">\n<input type=\"submit\"
value=\"Update\">\n</form>\n");

            adminHeader(req, resp, UPDATE_PROJ_JS);
            adminReply(req, resp, reply.toString());
            adminFooter(req, resp);
      }


}
```

UpdateProjectEntity.java

```
/*********************************************************************
 *
 * File     :     UpdateProjectEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a Project in the TPTeam
 *                database
 *
 *********************************************************************/

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateProjectEntity.java
 *
 * Description    :     Servlet that updates a Project in the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateProjectEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;
      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Updates a Project with the passed in form inputs
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
```

```
                  throws ServletException, IOException {
            String projId = req.getParameter("projId");
            String name = req.getParameter("projName");
            String description = req.getParameter("projDesc");
            String prodId = req.getParameter("prod");
            String[] userIds = req.getParameterValues("team");
            if(userIds == null)
            {
                  userIds = new String[0];
            }
            Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
            // For standalone debug
            //Session s =
HibernateUtil.getSessionFactory().getCurrentSession();
            Transaction tx = null;
            try {
                  tx = s.beginTransaction();
                  Project proj = (Project)s.load(Project.class, new
Integer(projId));
                  proj.setName(name);
                  proj.setDescription(description);
                  Product prod = (Product) s.load(Product.class, new
Integer(prodId));
                  proj.setProduct(prod);

      proj.getTpteamUsers().removeAll(proj.getTpteamUsers());
                  for (String userId : userIds) {
                        TpteamUser tpTeamUser = (TpteamUser)
s.load(TpteamUser.class,
                                    new Integer(userId));
                        // Project is not the inverse=true, so use it
to associate
                        // proj_user map table
                        proj.addToTpteamUsers(tpTeamUser);
                  }
                  s.flush();
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
            adminHeader(req, resp, null);
            String reply = "<h3>Update Project " + name + " was
Successful</h3>";
            adminReply(req, resp, reply);
            adminFooter(req, resp);
      }
}
```

UpdateTest.java

```
/*********************************************************************
 *
 * File     :     UpdateTest.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Test by first selecting its parent Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateTest.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam Test by first selecting its
 *                      parent Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateTest extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProjAvailable = false;

      protected String mProjRows = null;

      protected String mRemoteUser = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"updateTest2\"><th>Project</th><td>";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Update Test Tree\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
```

```
            super.init(config);
      }

      /**
       * Gathers Project select list form and renders so user
       * can select parent Project of Test to be updated
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mRemoteUser = req.getRemoteUser();
                  getProjRows();
                  if (mIsProjAvailable == false) {
                        StringBuffer error = new StringBuffer(
                                    "<h3>Error: No Project
Available</h3>");
                        throwError(req, resp, error, this);
                  } else {
                        StringBuffer reply = new StringBuffer(
                                    "<h4>Update Test Tree: Select
Parent Project</h4>\n");
                        reply.append("<table border=\"2\">" + mProjRows
+ "</table>");
                        showPage(req, resp, reply, null, this);
                  }
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Gets all TPTeam Projects and wraps them
       * into HTML select option tags
       *
       * @return The Project select option tags
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      protected String getProjRows() throws Exception {
            Session s = Activator.getDefault().getHiberSessionFactory()
                        .getCurrentSession();
            Transaction tx = null;
            List<Project> projs = null;
            StringBuffer projRows = new StringBuffer();
            try {

                  tx = s.beginTransaction();
```

```
                  projs = s.createQuery("from Project as p order by
p.name asc")
                              .list();
                  for (Project proj : projs) {
                        String desc = proj.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId()
                                    + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }

            if (projRows.length() > 0) {
                  mIsProjAvailable = true;
                  mProjRows = projRows.toString();
            }
            return mProjRows;
      }

}
```

UpdateTest2.java

```
/*********************************************************************
 *
 * File     :     UpdateTest2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Test by selecting it from the Test tree
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateTest2.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam Test by selecting it from the
 *                      Test tree
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class UpdateTest2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mProjID = null;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateTest3\">\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gets the ID of the Project selected and renders test tree for
       * Test selection
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mProjID = req.getParameter("projId");
                  getPage(req, resp);
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Helper method for rendering a Project's Test tree
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      protected void getPage(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply
                        .append("<h4>Select a Folder or Test, then
Click the Update Button</h4>\n");
            reply
                        .append("<form method=\"post\"
onSubmit=\"return validateForm(this);\">\n");

            reply.append("<table border=\"1\"><tr><td>\n");

            reply.append(ServletUtil.getTestTree(mProjID, false));

            reply.append(mFormAction);

            reply
                        .append("<input type=\"hidden\" name=\"testID\"
value=\"\">\n</td></tr></table><p>\n<input type=\"submit\"
value=\"Update\">\n</form>\n");

            showPage(req, resp, reply, ServletUtil.UPDATE_TEST_TREE_JS
                        + ServletUtil.ADD_TEST_TREE_JS +
ServletUtil.ADD_TEST_TREE_CSS, this);
      }

}
```

UpdateTest3.java

```
/*********************************************************************
 *
 * File     :     UpdateTest3.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that loads the details for a particular Test
 *                into a form so that a user can edit and update them
 *
 *********************************************************************/

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     UpdateTest3.java
 *
 * Description    :     Servlet that loads the details for a particular
 *                      Test into a form so that a user can edit and
 *                      update them
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateTest3 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTestID = null;

      protected String mJavaScript = null;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateTestEntity\">\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }


      /**
```

```
       * Gets the ID of the Test selected and renders the details
       * in a form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mTestID = req.getParameter("testID");
                  showUpdateTestPage3(req, resp);
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: " +
e.getMessage() + "<br>"
                              + e.getCause() + "</h3>");
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Helper method for rendering the form HTML
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      protected void showUpdateTestPage3(HttpServletRequest req,
                  HttpServletResponse resp) throws ServletException,
IOException,
                  Exception {

            StringBuffer reply = new StringBuffer("<h4>Update Test Plan
Node</h4>\n");
            reply.append("<form name=\"updateTest\" method=\"post\"
onSubmit=\"return validateForm(this);\">\n");
            reply.append("<table border=\"1\">\n");
            reply.append(getUpdateTestRows(mTestID));
            reply.append("</table><p>\n");
            reply.append("<input type=\"submit\"
value=\"Update\">\n</form>\n");

            showPage(req, resp, reply, mJavaScript, this);
      }

      /**
       * Loads the Test selected by the user so that
       * its details can be displayed
       *
       * @param testId the Test ID
       * @return the Test details as HTML form inputs
       * @throws Exception
```

```
       */
      protected String getUpdateTestRows(String testId) throws
Exception {
            Transaction tx = null;
            StringBuffer updateRows = new StringBuffer();
            try {
                  Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
                  tx = s.beginTransaction();
                  Test test = (Test) s.load(Test.class, new
Integer(testId));

                  // Set appropriate JavaScript validation script
                  if
(test.getIsFolder().toString().equalsIgnoreCase("Y")) {
                        mJavaScript =
ServletUtil.UPDATE_TEST_FOLDER_JS;
                  } else {
                        mJavaScript = ServletUtil.UPDATE_TEST_DEF_JS;
                  }

                  updateRows.append(getFolderUpdateRows(test));

                  if (test.getJunitTests().size() > 0) {
                        updateRows.append(getJUnitUpdateRows(test));
                  }
                  tx.commit();

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return updateRows.toString();
      }

      /**
       * Helper method that renders the HTML form input
       * elements for the JUnit details
       *
       * @param test the JUnit Test
       * @return the HTML form inputs
       */
      protected String getJUnitUpdateRows(Test test) {
            StringBuffer updateRows = new StringBuffer();
            for (JunitTest jUnitTest : test.getJunitTests()) {
                  updateRows.append("<tr><th>Eclipse
Home:</th><td><input type=\"text\" name=\"eclipseHome\" size=\"75\"
value=\""
                              + jUnitTest.getEclipseHome() +
"\"></td></tr>\n");
                  updateRows.append("<tr><th>Eclipse
Workspace:</th><td><input type=\"text\" name=\"eclipseWorkspace\"
size=\"75\" value=\""
                              + jUnitTest.getWorkspace() +
"\"></td></tr>\n");
```

```
                  updateRows.append("<tr><th>Eclipse
Project:</th><td><input type=\"text\" name=\"eclipseProj\" size=\"75\"
value=\""
                              + jUnitTest.getProject() +
"\"></td></tr>\n");
                  updateRows.append("<tr><th>Test Suite:</th><td><input
type=\"text\" name=\"testSuite\" size=\"75\" value=\""
                              + jUnitTest.getTestSuite() +
"\"></td></tr>\n");
                  updateRows.append("<tr><th>Report
Directory:</th><td><input type=\"text\" name=\"reportDir\" size=\"75\"
value=\""
                              + jUnitTest.getReportDir() +
"\"></td></tr>\n");
                  updateRows.append("<tr><th>TPTP Connection
URL:</th><td><input type=\"text\" name=\"tptpConn\" size=\"75\"
value=\""
                              + jUnitTest.getTptpConnection() +
"\"></td></tr>\n");
            }
            return updateRows.toString();
      }

      /**
       * Helper method that renders the HTML form input
       * elements for the core Test details
       *
       * @param test the Test
       * @return the HTML form inputs
       */
      protected String getFolderUpdateRows(Test test) {
            StringBuffer updateRows = new StringBuffer();
            String desc = "";
            if (test.getDescription() != null
                        && !test.getDescription().equalsIgnoreCase(""))
{
                  desc = test.getDescription();
            }
            updateRows.append("<tr><th>Name:</th><td>"
                        + "<input type=\"text\" name=\"testName\"
value=\"" + test.getName() + "\" size=\"75\">"
                        + "<input type=\"hidden\" name=\"testID\"
value=\"" + mTestID + "\"></td></tr>\n");

            updateRows.append(mFormAction);

            updateRows.append("<tr><th>Description:</th><td>"
                        + "<input type=\"text\" name=\"testDesc\"
value=\""
                        + desc + "\" size=\"75\"></td></tr>\n");

            return updateRows.toString();
      }

}
```

UpdateTestEntity.java

```
/*********************************************************************
 *
 * File     :     UpdateTestEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a Test in the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.Hashtable;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.tptp.TPTestCRUD;

/*********************************************************************
 * File           :     UpdateTestEntity.java
 *
 * Description    :     Servlet that updates a Test in the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class UpdateTestEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTestID = null;

      protected String mTestName = null;

      protected String mTestDesc = null;

      protected String mTestTypeName = null;

      protected String mRemoteUser = null;

      boolean mIsFolder = false;
```

```
      protected Test mTestStub = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Updates a Test with the passed in form inputs
       * Renders results, including errors, to the the user.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            mTestID = req.getParameter("testID");
            mTestName = req.getParameter("testName");
            mTestDesc = req.getParameter("testDesc");
            mRemoteUser = req.getRemoteUser();

            try {
                  updateTest(req, resp);
                  StringBuffer reply = new StringBuffer("<h3>Update
Test "
                              + mTestName + " was Successful</h3>");
                  showPage(req, resp, reply, null, this);
            } catch (Exception e) {
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  e.printStackTrace();
                  throwError(req, resp, error, this);
                  return;
            }
      }

      /**
       * Helper method that requests the actual
       * Test update.  Creates a test update request
       * TPEvent and requests that TPTestCRUD perform
       * the update.
       *
       * @see TPTestCRUD
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */

      protected void updateTest(HttpServletRequest req,
HttpServletResponse resp)
                  throws Exception {
            mTestStub = new Test();
            mTestStub.setId(Integer.parseInt(mTestID));
```

```
            mTestStub.setName(mTestName);
            mTestStub.setDescription(mTestDesc);
            if (req.getParameter("eclipseHome") != null
                        && req.getParameter("eclipseHome").length() > 0
                        && !req.getParameter("eclipseHome").equals(""))
{
                  JunitTest junit = new JunitTest();

      junit.setEclipseHome(req.getParameter("eclipseHome"));

      junit.setWorkspace(req.getParameter("eclipseWorkspace"));
                  junit.setProject(req.getParameter("eclipseProj"));
                  junit.setTestSuite(req.getParameter("testSuite"));
                  junit.setReportDir(req.getParameter("reportDir"));

      junit.setTptpConnection(req.getParameter("tptpConn"));
                  mTestStub.addJunitTest(junit);
            }
            String testXML = TestXML.getXML(mTestStub);

            // Wrap info into TPEvent
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.ID_KEY, String.valueOf(mTestID));
            dictionary.put(TPEvent.SEND_TO, Activator.getDefault()
                        .getTPBridgeClient().getTPMgrECFID());
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());
            dictionary.put(TPEvent.TEST_XML_KEY, testXML);

            TPEvent tpEvent = new
TPEvent(ITPBridge.TEST_UPDATE_REQ_TOPIC,
                        dictionary);

            TPTestCRUD.sendTestUpdateResponse(tpEvent);
      }
}
```

UpdateUser.java

```
/**********************************************************************
 *
 * File     :     UpdateUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam User
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     UpdateUser.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam User
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class UpdateUser extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsUserAvailable = false;

      protected String mUserRows = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"updateUser2\"><td>";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"userId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"Update\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
      /**
       * Gathers the User selection list, renders input form
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mUserRows = getUserRows();
                  if (mIsUserAvailable == false) {
                        StringBuffer error = new StringBuffer(
                                    "<h3>Error: No Users Exist</h3>");
                        throwError(req, resp, error, this);
                  } else {
                        StringBuffer reply = new StringBuffer(
                                    "<h4>Update User</h4>\n<table
border=\"2\">\n<th>Last, First</th><th></th>\n"
                                                + mUserRows +
"</table>");
                        showPage(req, resp, reply, null, this);
                  }
            } catch (Exception e) {
                  String error = "<h3>Error: " + e.getMessage() +
"<br>"
                              + e.getCause() + "</h3>";
                  adminError(req, resp, error);
                  return;
            }
      }

      /**
       * Gets all TPTeam Users and wraps them in HTML
       * selection list option tags
       *
       * @return the HTML String User option tag
       * @throws Exception
       */
      @SuppressWarnings("unchecked")
      protected String getUserRows() throws Exception {
            Session s = Activator.getDefault().getHiberSessionFactory()
                        .getCurrentSession();
            // For standalone
            // Session s =
HibernateUtil.getSessionFactory().getCurrentSession();

            Transaction tx = null;
            List<TpteamUser> users = null;
            StringBuffer userRows = new StringBuffer();
            try {

                  tx = s.beginTransaction();

                  users = s.createQuery(
```

```
                              "from TpteamUser as user order by
user.lastName asc")
                              .list();
                  for (TpteamUser user : users) {
                        userRows.append(rowNameHeader +
user.getLastName() + ", "
                                    + user.getFirstName() + "</td>\n");
                        userRows.append(rowIDHeader + " value=\"" +
user.getId()
                                    + "\">\n");
                        userRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (userRows.length() > 0)
                  mIsUserAvailable = true;

            return userRows.toString();
      }
}
```

UpdateUser2.java

```
/*********************************************************************
 *
 * File     :     UpdateUser2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the details of a particular
 *                TPTeam user in a form for editable updating
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.hibernate.Session;
import org.hibernate.Transaction;

import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Role;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;

/*********************************************************************
 * File           :     UpdateUser2.java
 *
 * Description    :     Servlet that displays the details of a
 *                      particular TPTeam user in a form for editable
 *                      updating
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*********************************************************************/
public class UpdateUser2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateUserEntity\">\n";

      protected String mUserId = null;

      protected String mFirstName = null;

      protected String mLastName = null;
```

```
	protected String mUserName = null;

	protected String mPassword = null;

	protected String mECFId = null;

	protected String mEmail = null;

	protected String mPhone = null;

	protected String mRoleId = null;

	protected String mRoleOptions = null;

	protected boolean mIsRoleAvailable;

	public void init(ServletConfig config) throws ServletException {
		super.init(config);
	}

	/**
	 * Gathers the details of TPTeam user selected and the Role
	 * selection list, renders input form
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			if(this instanceof UserServlet)
			{
				mUserId =
String.valueOf(getRemoteUserID(req.getRemoteUser()));
				getUser();
			}
			else
			{
				mUserId = req.getParameter("userId");
				getUser();
				getRoleOptions();
			}
			getPage(req, resp);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Loads a TPTeam user from the database, populates
```

```
	 * member variables for setting form defaults
	 *
	 * @throws Exception
	 */
	public void getUser() throws Exception {
		TpteamUser user = null;
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		Transaction tx = null;
		try {

			tx = s.beginTransaction();
			user = (TpteamUser) s.load(TpteamUser.class, new
Integer(mUserId));
			mFirstName = user.getFirstName();
			mLastName = user.getLastName();
			mUserName = user.getUserName();
			mPassword = user.getPassword();
			mECFId = user.getEcfId();
			mEmail = user.getEmail();
			mPhone = user.getPhone();
			mRoleId = String.valueOf(user.getRole().getRoleId());

			tx.commit();
		} catch (Exception e) {
			if (tx != null)
				tx.rollback();
			throw e;
		}
	}

	/**
	 * Helper method that gets all Roles and wraps
	 * them in HTML option tags
	 *
	 * @return The Product option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getRoleOptions() throws Exception {
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		// For standalone
		// Session s =
HibernateUtil.getSessionFactory().getCurrentSession();

		Transaction tx = null;
		List<Role> roles = null;
		StringBuffer roleOptions = new StringBuffer();
		try {

			tx = s.beginTransaction();

			roles = s.createQuery("from Role as role order by
role.name asc")
					.list();
			for (Role role : roles) {
```

```
                        if (role.getRoleId() ==
Integer.parseInt(mRoleId)) {
                              roleOptions.append("<option value=\"" +
role.getRoleId()
                                    + "\" selected>" +
role.getName() + "</option>\n");
                        } else {
                              roleOptions.append("<option value=\"" +
role.getRoleId()
                                    + "\">" + role.getName() +
"</option>\n");
                        }
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (roleOptions.length() > 0)
                  mIsRoleAvailable = true;

            mRoleOptions = roleOptions.toString();
            return roleOptions.toString();
      }

      /**
       * Helper method to render the HTML page, including
       * form inputs, JavaScript
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
      protected void getPage(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply.append("<h4>Update User</h4>\n");
            reply
                        .append("<form method=\"post\"
onSubmit=\"return validateForm(this);\">\n");
            reply
                        .append("<table ><tr><th>Last
Name:</th><td><input type=\"text\" name=\"lastName\"  value=\""
                              + mLastName + "\"
size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>First Name:</th><td><input
type=\"text\" name=\"firstName\"  value=\""
                              + mFirstName + "\"
size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>User Name:</th><td><input
type=\"text\" name=\"userName\"  value=\""
```

```
                                    + mUserName + "\"
size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>Password:</th><td><input
type=\"password\" name=\"password\" value=\""
                                    + mPassword + "\"
size=\"50\"></td></tr>\n");
            reply
                        .append("<tr><th>Password
(confirm):</th><td><input type=\"password\" name=\"passwordConfirm\"
value=\""
                                    + mPassword + "\"
size=\"50\"></td></tr>\n");
            reply
                        .append("<tr><th>ECF Id:</th><td><input
type=\"text\" name=\"ecfID\"  value=\""
                                    + mECFId + "\"
size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>Email:</th><td><input
type=\"text\" name=\"email\"  value=\""
                                    + mEmail + "\"
size=\"25\"></td></tr>\n");
            reply
                        .append("<tr><th>Phone (ddd-ddd-
dddd):</th><td><input type=\"text\" name=\"phone\"  value=\""
                                    + mPhone + "\"
size=\"25\"></td></tr>\n");
            if(!(this instanceof UserServlet))
            {
                  reply.append("<tr><th>Role:</th><td><select
name=\"role\">"
                        + mRoleOptions + "</select></td></tr>\n");
            }
            reply
                        .append("</table>\n<br>\n<input type=\"hidden\"
name=\"userId\" value=\""
                                    + mUserId
                                    + "\">\n"
                                    + mFormAction
                                    + "<input type=\"submit\"
value=\"Update\">\n</form>\n");

            showPage(req, resp, reply, UPDATE_USER_JS, this);
      }
}
```

UpdateUserEntity.java

```
/*********************************************************************
 *
 * File     :     UpdateUserEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a user in the TPTeam
 *                database
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update;

import java.io.IOException;
import java.util.Date;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Role;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;

/*********************************************************************
 * File           :     UpdateUserEntity.java
 *
 * Description    :     Servlet that updates a user in the TPTeam
 *                      database
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class UpdateUserEntity extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Updates a TPTeam user with the passed in form inputs
       * Renders results, including errors, to the the browser.
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
```

```
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            String userId = req.getParameter("userId");
            String firstName = req.getParameter("firstName");
            String lastName = req.getParameter("lastName");
            String userName = req.getParameter("userName");
            String password = req.getParameter("password");
            String email = req.getParameter("email");
            String ecfId = req.getParameter("ecfID");
            String phone = req.getParameter("phone");
            String roleId = req.getParameter("role");
            Transaction tx = null;
            try {
                  int remoteUserId =
ServletUtil.getRemoteUserID(req.getRemoteUser());
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = (TpteamUser)
s.load(TpteamUser.class,
                              new Integer(userId));
                  user.setFirstName(firstName);
                  user.setLastName(lastName);
                  user.setUserName(userName);
                  user.setEmail(email);
                  user.setPhone(phone);
                  user.setEcfId(ecfId);
                  user.setModifiedBy(remoteUserId);
                  user.setModifiedDate(new Date());
                  if (!password.equalsIgnoreCase(user.getPassword())) {

      user.setPassword(ServletUtil.getSHA1Hash(password));
                  }

                  if(!(this instanceof UserServlet))
                  {
                        Role role = (Role) s.load(Role.class, new
Integer(roleId));
                        user.setRole(role);
                  }
                  tx.commit();

                  StringBuffer reply = new StringBuffer("<h3>Update
TPTeam User "
                              + lastName + ", " + firstName + " was
Successful</h3>");
                  showPage(req, resp, reply, null, this);

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  StringBuffer error = new StringBuffer("<h3>Error: "
                              + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                  throwError(req, resp, error, this);
```

```
                    return;
                }
            }
        }
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view

ViewProduct.java

```
/*********************************************************************
 *
 * File     :     ViewProduct.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for selecting
 *                a particular TPTeam Product for viewing
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Product;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     ViewProduct.java
 *
 * Description    :     Servlet that displays an input form for
 *                      selecting a particular TPTeam Product for
 *                      viewing
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ViewProduct extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProdAvailable = false;

      protected String mProdRows = null;

      protected String mRowHeader =
"<tr><th>Name</th><th>Description</th></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
	/**
	 * Renders the new Product input form
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			getProdRows();
			if (mIsProdAvailable == false) {
				StringBuffer error = new StringBuffer(
						"<h3>Error: No Product
Available</h3>");
				throwError(req, resp, error, this);
			} else {
				StringBuffer reply = new StringBuffer(
						"<h4>View Products</h4>\n<table
border=\"1\">"
								+ mRowHeader +
mProdRows + "</table>");
				showPage(req, resp, reply, null, this);
			}
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Helper method that gets all Products and wraps
	 * them in HTML option tags
	 *
	 * @return The Product option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	public String getProdRows() throws Exception {
		Session s = null;
		if (Activator.getDefault() != null) {
			s = Activator.getDefault().getHiberSessionFactory()
					.getCurrentSession();
		} else {
			// For standalone outside OSGi
			s =
HibernateUtil.getSessionFactory().getCurrentSession();
		}

		Transaction tx = null;
		List<Product> prods = null;
		StringBuffer prodRows = new StringBuffer();
		try {
```

```
                  tx = s.beginTransaction();

                  prods = s.createQuery("from Product as p order by
p.name asc")
                              .list();
                  for (Product prod : prods) {
                        String desc = prod.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        prodRows.append("<tr><td>" + prod.getName() +
"</td>");
                        prodRows.append("<td>" + desc +
"</td></tr>\n");
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (prodRows.length() > 0)
                  mIsProdAvailable = true;
            mProdRows = prodRows.toString();
            return mProdRows;
      }
}
```

ViewProject.java

```
/**********************************************************************
 *
 * File     :     ViewProject.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a particular TPTeam Project
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     ViewProject.java
 *
 * Description    :     Servlet that displays an input form for viewing
 *                      a particular TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class ViewProject extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProjAvailable = false;

      protected String mProjRows = null;

      protected String mRemoteUser = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"viewProject2\"><th>Project</th><td>";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"View\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
```

```
	}

	/**
	 * Renders the update Project input form for selecting one
Project
	 *
	 * @param req
	 *            The Servlet Request
	 * @param resp
	 *            The Servlet Response
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mRemoteUser = req.getRemoteUser();
			getProjRows();
			if (mIsProjAvailable == false) {
				StringBuffer error = new StringBuffer(
						"<h3>Error: No Project
Available</h3>");
				throwError(req, resp, error, this);
			} else {
				StringBuffer reply = new StringBuffer(
						"<h4>View Project</h4>\n<table
border=\"2\">"
								+ mProjRows +
"</table>");
				showPage(req, resp, reply, null, this);
			}
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Helper method that gets all Project and wraps them in HTML
option tags
	 *
	 * @return The Product option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getProjRows() throws Exception {
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		Transaction tx = null;
		List<Project> projs = null;
		StringBuffer projRows = new StringBuffer();
		try {

			tx = s.beginTransaction();
```

```
                  projs = s.createQuery("from Project as p order by
p.name asc")
                              .list();
                  for (Project proj : projs) {
                        projRows.append(rowNameHeader +
proj.getName());
                        if (proj.getDescription() != null
                                    &&
!proj.getDescription().equalsIgnoreCase("null")) {
                              projRows.append(": " +
proj.getDescription());
                        }
                        projRows.append("</td>\n" + rowIDHeader + "
value=\""
                                    + proj.getId() + "\">\n" +
rowSubmitHeader + "\n");
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;

            mProjRows = projRows.toString();

            return mProjRows;
      }
}
```

ViewProject2.java

```
/*********************************************************************
 *
 * File     :     ViewProject2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the view details
 *                a particular TPTeam Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.Set;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     ViewProject2.java
 *
 * Description    :     Servlet that displays the view details
 *                      a particular TPTeam Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ViewProject2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTeam = null;

      protected String mProjId = null;

      protected String mProjName = null;

      protected String mProjDesc = null;

      protected String mProd = null;

      protected int[] mUserIds = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
	/**
	 * Gathers the Project selected, the Product, and User
	 * lists, renders the details
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mProjId = req.getParameter("projId");
			getProject();
			getPage(req, resp);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Loads a Project from the database, populates
	 * member variables for TPTeam user and Product IDs
	 *
	 * @throws Exception
	 */
	protected void getProject() throws Exception {
		Project proj = null;
		Session s = Activator.getDefault().getHiberSessionFactory()
				.getCurrentSession();
		Transaction tx = null;
		try {

			tx = s.beginTransaction();
			proj = (Project) s.load(Project.class, new
Integer(mProjId));
			if (proj.getDescription() != null) {
				mProjDesc = proj.getDescription();
			} else {
				mProjDesc = "";
			}
			mProjName = proj.getName();
			mProd = proj.getProduct().getName();
			getTeam(proj.getTpteamUsers());
			s.flush();
			tx.commit();
		} catch (Exception e) {
			if (tx != null)
				tx.rollback();
			throw e;
		}
```

```
	}

	/**
	 * Gets all TPTeam users associated with the Project
	 *
	 * @param team The TPTeam users
	 * @throws Exception
	 */
	public void getTeam(Set<TpteamUser> team) throws Exception {
		StringBuffer teamUsers = new StringBuffer();
		if (team.size() < 1) {
			mTeam = "";
			return;
		}
		for (TpteamUser user : team) {
			teamUsers.append(user.getLastName() + ", " +
user.getFirstName()
					+ " (" + user.getUserName() + ")<br>\n");
		}
		mTeam = teamUsers.toString();
	}

	/**
	 * Helper method to render the HTML page
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws ServletException
	 * @throws IOException
	 */
	protected void getPage(HttpServletRequest req,
HttpServletResponse resp)
			throws ServletException, IOException, Exception {
		StringBuffer reply = new StringBuffer();
		reply.append("<h4>View Project</h4>\n");
		reply
				.append("<table border=\"1\"><tr><th
align=\"left\">Name:</th><td align=\"right\">"
					+ mProjName + "</td></tr>\n");
		reply
				.append("<tr><th
align=\"left\">Description:</th><td align=\"right\">"
					+ mProjDesc + "</td></tr>\n");
		reply.append("<tr><th align=\"left\">Product:</th><td
align=\"right\">"
				+ mProd + "</td></tr>\n");
		reply
				.append("<tr><th align=\"left\">Team
Members:</th><td align=\"right\">"
					+ mTeam + "</td></tr>\n");
		reply
				.append("<tr><th align=\"left\">View  Test
Tree:</th><td align=\"right\"><form method=\"post\"
action=\"viewTest2\">\n<input type=\"hidden\" name=\"projId\" value=\""
					+ mProjId
					+ "\">\n<input type=\"submit\"
value=\"Test Tree Details\">\n</form>\n");
```

```
            reply.append("</table>\n<br>\n");

            showPage(req, resp, reply, null, this);
      }

      /**
       * Getter for TPTeam Project members in HTML format
       *
       * @return team members in html format
       */
      public String getTeam()
      {
            return mTeam;
      }

}
```

ViewTest.java

```
/*********************************************************************
 *
 * File     :     ViewTest.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a TPTeam Test by first selecting its parent Project
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;

/*********************************************************************
 * File           :     ViewTest.java
 *
 * Description    :     Servlet that displays an input form for viewing
 *                      a TPTeam Test by first selecting its parent
 *                      Project
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ViewTest extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsProjAvailable = false;

      protected String mProjRows = null;

      protected String mRemoteUser = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"viewTest2\"><th align=\"left\">Project</th><td
align=\"right\">";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"projId\"";
```

```
	protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"View Test Tree\"></td>\n</form></tr>\n";

	public void init(ServletConfig config) throws ServletException {
		super.init(config);
	}

	/**
	 * Gathers Project select list form and renders so user
	 * can select parent Project of Test to be updated
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mRemoteUser = req.getRemoteUser();
			getProjRows();
			if (mIsProjAvailable == false) {
				StringBuffer error = new StringBuffer(
						"<h3>Error: No Project
Available</h3>");
				throwError(req, resp, error, this);
			} else {
				StringBuffer reply = new StringBuffer(
					"<h4>View Test Tree: Select Parent
Project</h4>\n<table border=\"2\">"
							+ mProjRows +
"</table>");
				showPage(req, resp, reply, null, this);
			}
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Gets all TPTeam Projects and wraps them
	 * into HTML select option tags
	 *
	 * @return The Project select option tags
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getProjRows() throws Exception {
		Transaction tx = null;
		Set<Project> projs = new HashSet<Project>(0);
		StringBuffer projRows = new StringBuffer();
```

```
            int remoteUserId = -1;
            try {
                  if (this instanceof UserServlet) {
                        remoteUserId = getRemoteUserID(mRemoteUser);
                  }

                  Session s =
Activator.getDefault().getHiberSessionFactory()
                        .getCurrentSession();
                  tx = s.beginTransaction();

                  if (remoteUserId == -1) {
                        List<Project> projList = s.createQuery(
                                    "from Project as p order by p.name
asc").list();
                        projs.addAll(projList);
                  } else {
                        TpteamUser user = (TpteamUser) s.createQuery(
                                    "from TpteamUser as user where
user.id = "
                                                +
remoteUserId).uniqueResult();
                        projs = user.getProjects();
                  }

                  for (Project proj : projs) {
                        String desc = proj.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId()
                                    + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;
            mProjRows = projRows.toString();
            return mProjRows;
      }
}
```

ViewTest2.java

```
/*********************************************************************
 *
 * File     :     ViewTest2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a TPTeam Test by selecting it from the Test tree
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     ViewTest2.java
 *
 * Description    :     Servlet that displays an input form for viewing
 *                      a TPTeam Test by selecting it from the Test
 *                      tree
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ViewTest2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mProjID = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gets the ID of the Project selected and renders test tree for
       * Test selection
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws IOException
       * @throws ServletException
       */
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                  throws ServletException, IOException {
            try {
                  mProjID = req.getParameter("projId");
```

```
                getPage(req, resp);
            } catch (Exception e) {
                StringBuffer error = new StringBuffer("<h3>Error: "
                            + e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
                throwError(req, resp, error, this);
                return;
            }
      }

      /**
       * Helper method for rendering a Project's Test tree
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       * @throws Exception
       */
      protected void getPage(HttpServletRequest req,
HttpServletResponse resp)
                throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply
                        .append("<h4>Select a Folder or Test, then
Click the Details Button</h4>\n");
            reply
                        .append("<form name=\"viewTest\"
method=\"post\" onSubmit=\"return validateForm(this);\">\n");

            reply.append("<table border=\"1\"><tr><td>\n");

            reply.append(ServletUtil.getTestTree(mProjID, false));

            reply
                        .append("<input type=\"hidden\" name=\"testID\"
value=\"\">\n</td></tr></table><p>\n<input type=\"submit\"
value=\"Details\">\n</form>\n");

            showPage(req, resp, reply, ServletUtil.VIEW_TEST_TREE_JS
                        + ServletUtil.ADD_TEST_TREE_JS +
ServletUtil.ADD_TEST_TREE_CSS,
                        this);
      }
}
```

ViewTest3.java

```
/*********************************************************************
 *
 * File     :     ViewTest3.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that loads the details for a particular Test
 *                so that a user can view them
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.text.ParseException;
import java.util.Set;
import java.util.TreeSet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TestExecution;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/*********************************************************************
 * File           :     ViewTest3.java
 *
 * Description    :     Servlet that loads the details for a particular
 *                      Test so that a user can view them
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class ViewTest3 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mTestID = null;

      protected String mJavaScript = null;

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }

      /**
       * Gets the ID of the Test selected and renders the details
       * in HTML
       *
```

```
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mTestID = req.getParameter("testID");
			showUpdateTestPage3(req, resp);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: " +
e.getMessage() + "<br>"
					+ e.getCause() + "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Helper method for rendering the test details HTML
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws ServletException
	 * @throws IOException
	 * @throws Exception
	 */
	protected void showUpdateTestPage3(HttpServletRequest req,
			HttpServletResponse resp) throws ServletException,
IOException,
			Exception {

		StringBuffer reply = new StringBuffer();
		reply.append("<h4>View Test Plan Node</h4>\n");

		reply.append("<table border=\"1\">\n");
		reply.append(getTestRows(mTestID));
		reply.append("</table><p>\n");

		showPage(req, resp, reply, null, this);
	}

	/**
	 * Loads the Test selected by the user so that
	 * its details can be displayed
	 *
	 * @param testId the Test ID
	 * @return the Test details as HTML
	 * @throws Exception
	 */
	protected String getTestRows(String testId) throws Exception {
		Transaction tx = null;
		StringBuffer updateRows = new StringBuffer();
		try {
```

```
                  Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
                  tx = s.beginTransaction();
                  Test test = (Test) s.load(Test.class, new
Integer(testId));
                  updateRows.append(getFolderRows(test));

                  if (test.getJunitTests().size() > 0) {
                        updateRows.append(getJUnitUpdateRows(test));
                  }

                  if(test.getIsFolder() == 'N' || test.getIsFolder() ==
'n')
                  {
                        updateRows.append("<tr><th
align=\"left\">Execution Results:</th><td align=\"right\">"
                                    +
getTestResults(test.getTestExecutions()) + "</td></tr>\n");
                  }

                  tx.commit();

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return updateRows.toString();
      }

      /**
       * Helper method that renders the HTML
       * for the JUnit details
       *
       * @param test the JUnit Test
       * @return the Test details in HTML
       */
      protected String getJUnitUpdateRows(Test test) {
            StringBuffer updateRows = new StringBuffer();
            for (JunitTest jUnitTest : test.getJunitTests()) {
                  updateRows.append( "<tr><th align=\"left\">Eclipse
Home:</th><td align=\"right\">"
                                    + jUnitTest.getEclipseHome() +
"</td></tr>\n");
                  updateRows.append( "<tr><th align=\"left\">Eclipse
Workspace:</th><td align=\"right\">"
                                    + jUnitTest.getWorkspace() +
"</td></tr>\n");
                  updateRows.append( "<tr><th align=\"left\">Eclipse
Project:</th><td align=\"right\">"
                                    + jUnitTest.getProject() +
"</td></tr>\n");
                  updateRows.append( "<tr><th align=\"left\">Test
Suite:</th><td align=\"right\">"
                                    + jUnitTest.getTestSuite() +
"</td></tr>\n");
```

```
                  updateRows.append( "<tr><th align=\"left\">Report
Directory:</th><td align=\"right\">"
                              + jUnitTest.getReportDir() +
"</td></tr>\n");
                  updateRows.append( "<tr><th align=\"left\">TPTP
Connection URL:</th><td align=\"right\">"
                              + jUnitTest.getTptpConnection() +
"</td></tr>\n");
            }
            return updateRows.toString();
      }

      /**
       * Helper method that renders the HTML
       * for the core Test details
       *
       * @param test the Test
       * @return the core Test details as HTML
       */
      protected String getFolderRows(Test test) throws ParseException {
            StringBuffer updateRows = new StringBuffer();
            String desc = "";
            if (test.getDescription() != null
                        && !test.getDescription().equalsIgnoreCase(""))
{
                  desc = test.getDescription();
            }
            updateRows.append( "<tr><th align=\"left\">Name:</th><td
align=\"right\">"
                         + test.getName()  + "</td></tr>\n");

            updateRows.append( "<tr><th
align=\"left\">Description:</th><td align=\"right\">"
                        + desc + "</td></tr>\n");

            updateRows.append( "<tr><th align=\"left\">Created
By:</th><td align=\"right\">"
                         + test.getCreatedBy().getLastName() + ", " +
test.getCreatedBy().getFirstName()
                         + " (" + test.getCreatedBy().getUserName() +
")"+ "</td></tr>\n");

            updateRows.append( "<tr><th align=\"left\">Creation
Date:</th><td align=\"right\">"
                        +
TIMESTAMP_FORMAT.format(test.getCreatedDate()) + "</td></tr>\n");

            String modUser = "";
            String modDate = "";
            if(test.getModifiedBy() != null)
            {
                  modUser = test.getModifiedBy().getLastName() + ", " +
test.getModifiedBy().getFirstName()
                   + " (" + test.getModifiedBy().getUserName() + ")";
                  modDate =
TIMESTAMP_FORMAT.format(test.getModifiedDate()).toString();
            }
```

```
            updateRows.append( "<tr><th align=\"left\">Modified
By:</th><td align=\"right\">"
                        + modUser + "</td></tr>\n");

            updateRows.append( "<tr><th align=\"left\">Modification
Date:</th><td align=\"right\">"
                       + modDate + "</td></tr>\n");

            return updateRows.toString();
      }

      /**
       * Helper method that gets all of the selected Test's
       * execution results in HTML format
       *
       * @param testExecs the Test's Set of TestExecutions
       * @return the TestExecution details in HTML format
       */
      protected String getTestResults(Set<TestExecution> testExecs)
      {
            StringBuffer results = new StringBuffer();
            String resultRows = "";
            String execDate = "";
            String execBy = "";
            if(testExecs.size() < 1)
            {
                  return resultRows;
            }
            TreeSet<TestExecution> testExecTree = new
TreeSet<TestExecution>(testExecs);
            for(TestExecution exec : testExecTree)
            {
                  String status = "<b><font
color=\"green\">PASS</font></b>";
                  if(exec.getStatus() == 'F' || exec.getStatus() ==
'f')
                  {
                        status = "<b><font
color=\"red\">FAIL</font></b>";
                  }
                  execDate =
TIMESTAMP_FORMAT.format(exec.getExecDate());
                  execBy = exec.getTpteamUser().getLastName() + ", " +
exec.getTpteamUser().getFirstName()
                   + " (" + exec.getTpteamUser().getUserName() + ")";
                  results.append(status + ", " + execDate + ", " +
execBy + "<br>\n");
            }
            return results.toString();
      }
}
```

ViewUser.java

```
/**********************************************************************
 *
 * File     :     ViewUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a TPTeam User
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.List;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;

/**********************************************************************
 * File           :     ViewUser.java
 *
 * Description    :     Servlet that displays an input form for viewing
 *                      a TPTeam User
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class ViewUser extends ServletUtil {
      private static final long serialVersionUID =
7456848419577223441L;

      protected boolean mIsUserAvailable = false;

      protected String mUserRows = null;

      protected String rowNameHeader = "<tr><form method=\"post\"
action=\"viewUser2\"><td>";

      protected String rowIDHeader = "<input type=\"hidden\"
name=\"userId\"";

      protected String rowSubmitHeader = "<td><input type=\"submit\"
value=\"View\"></td>\n</form></tr>\n";

      public void init(ServletConfig config) throws ServletException {
            super.init(config);
      }
```

```
	/**
	 * Gathers the User selection list, renders input form
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mUserRows = getUserRows();
			if (mIsUserAvailable == false) {
				StringBuffer error = new
StringBuffer("<h3>Error: No Users Exist</h3>");
				throwError(req, resp, error, this);
			} else {
				StringBuffer reply = new StringBuffer("<h4>View
User</h4>\n<table border=\"2\">\n<th>Last, First</th><th></th>\n"
					+ mUserRows + "</table>");
				showPage(req, resp, reply, null, this);
			}
		} catch (Exception e) {
			String error = "<h3>Error: " + e.getMessage() +
"<br>"
					+ e.getCause() + "</h3>";
			adminError(req, resp, error);
			return;
		}
	}

	/**
	 * Gets all TPTeam Users and wraps them in HTML
	 * selection list option tags
	 *
	 * @return the HTML String User option tag
	 * @throws Exception
	 */
	@SuppressWarnings("unchecked")
	protected String getUserRows() throws Exception {
		 Session s =
Activator.getDefault().getHiberSessionFactory().getCurrentSession();
		Transaction tx = null;
		List<TpteamUser> users = null;
		StringBuffer userRows = new StringBuffer();
		try {

			tx = s.beginTransaction();

			users = s.createQuery(
					"from TpteamUser as user order by
user.lastName asc")
					.list();
			for (TpteamUser user : users) {
				userRows.append(rowNameHeader +
user.getLastName() + ", "
```

```
                                    + user.getFirstName() + "</td>\n");
                        userRows.append(rowIDHeader + " value=\"" +
user.getId()
                                    + "\">\n");
                        userRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (userRows.length() > 0)
                  mIsUserAvailable = true;

            return userRows.toString();
      }
}
```

ViewUser2.java

```
/**********************************************************************
 *
 * File     :     ViewUser2.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the details of a particular
 *                TPTeam user
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view;

import java.io.IOException;
import java.util.Set;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;

/**********************************************************************
 * File           :     ViewUser2.java
 *
 * Description    :     Servlet that displays the details of a
 *                      particular TPTeam user in a form for editable
 *                      updating
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class ViewUser2 extends ServletUtil {

      private static final long serialVersionUID =
7456848419577223441L;

      protected String mRemoteUser = null;

      protected String mUserId = null;

      protected String mFirstName = null;

      protected String mLastName = null;

      protected String mUserName = null;

      protected String mPassword = null;
```

```
	protected String mECFId = null;

	protected String mEmail = null;

	protected String mPhone = null;

	protected String mRole = null;

	protected String mProjects = null;

	public void init(ServletConfig config) throws ServletException {
		super.init(config);
	}

	/**
	 * Gathers the details of TPTeam user selected, renders HTML
	 *
	 * @param req The Servlet Request
	 * @param resp The Servlet Response
	 * @throws IOException
	 * @throws ServletException
	 */
	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		try {
			mRemoteUser = req.getRemoteUser();
			mUserId = req.getParameter("userId");
			getUser();
			showPage(req, resp);
		} catch (Exception e) {
			StringBuffer error = new StringBuffer("<h3>Error: "
					+ e.getMessage() + "<br>" + e.getCause()
+ "</h3>");
			throwError(req, resp, error, this);
			return;
		}
	}

	/**
	 * Loads a TPTeam user from the database, populates
	 * member variables for viewing details
	 *
	 * @throws Exception
	 */
	public TpteamUser getUser() throws Exception {
		TpteamUser user = null;
		Transaction tx = null;
		try {
			if (this instanceof UserServlet) {
				mUserId =
String.valueOf(getRemoteUserID(mRemoteUser));
			}
			Session s = null;
			if (Activator.getDefault() != null) {
				s =
Activator.getDefault().getHiberSessionFactory()
```

```
                                    .getCurrentSession();
                  } else {
                        // For standalone
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }

                  tx = s.beginTransaction();
                  user = (TpteamUser) s.load(TpteamUser.class, new
Integer(mUserId));
                  mFirstName = user.getFirstName();
                  mLastName = user.getLastName();
                  mUserName = user.getUserName();
                  mECFId = user.getEcfId();
                  mEmail = user.getEmail();
                  mPhone = user.getPhone();
                  mRole = user.getRole().getName();
                  getProjects(user.getProjects());
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            return user;
      }


      /**
       * Helper method to get all Projects associated with
       * the selected TPTeam user
       *
       * @param projects the Projects
       * @throws Exception
       */
      protected void getProjects(Set<Project> projects) throws
Exception {
            StringBuffer projs = new StringBuffer();
            if (projects == null || projects.size() < 1) {
                  mProjects = "";
                  return;
            }
            for (Project proj : projects) {
                  projs.append(proj.getName() + "<br>");
            }
            mProjects = projs.toString();
      }

      /**
       * Helper method to render the HTML page
       *
       * @param req The Servlet Request
       * @param resp The Servlet Response
       * @throws ServletException
       * @throws IOException
       */
```

```
      protected void showPage(HttpServletRequest req,
HttpServletResponse resp)
                  throws ServletException, IOException, Exception {
            StringBuffer reply = new StringBuffer();
            reply.append("<h4>View User</h4>\n");
            reply
                        .append("<table border=\"1\"><tr><th
align=\"left\">Last Name:</th><td align=\"right\">"
                                    + mLastName + "</td></tr>\n");
            reply
                        .append("<tr><th align=\"left\">First
Name:</th><td align=\"right\">"
                                    + mFirstName + "</td></tr>\n");
            reply
                        .append("<tr><th align=\"left\">User
Name:</th><td align=\"right\">"
                                    + mUserName + "</td></tr>\n");
            reply.append("<tr><th align=\"left\">ECF Id:</th><td
align=\"right\">"
                        + mECFId + "</td></tr>\n");
            reply.append("<tr><th align=\"left\">Email:</th><td
align=\"right\">"
                        + mEmail + "</td></tr>\n");
            reply
                        .append("<tr><th align=\"left\">Phone (ddd-ddd-
dddd):</th><td align=\"right\">"
                                    + mPhone + "</td></tr>\n");
            reply.append("<tr><th align=\"left\">Role:</th><td
align=\"right\">"
                        + mRole + "</td></tr>\n");
            reply
                        .append("<tr><th
align=\"left\">Projects:</th><td align=\"right\">"
                                    + mProjects + "</td></tr>\n");
            reply.append("</table>\n");

            showPage(req, resp, reply, null, this);
      }

      /** Setter, primarily used for test harness
       *
       * @param userId
       */
      public void setUserId(String userId)
      {
            mUserId = userId;
      }
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.add

AddTestByUser.java

```
/********************************************************************
 *
 * File     :     AddTestByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays input forms for creating
 *                a new TPTeam Test by a non-administrative Web user
 *
 ********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.add;

import java.util.Set;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddTest;

/********************************************************************
 * File           :     AddTestByUser.java
 *
 * Description    :     Servlet that displays input forms for creating
 *                      a new TPTeam Test by a non-administrative Web
 *                      user
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 ********************************************************************/
public class AddTestByUser extends AddTest implements UserServlet {
      private static final long serialVersionUID = 1L;

      /**
       * Gets all TPTeam Projects that the TPTeam user is
       * associated with and wraps them  into HTML
       * select option tags
       *
       * @return The Project select option tags
       * @throws Exception
       */
      protected String getProjOptions() throws Exception {
            Transaction tx = null;
            Set<Project> projs = null;
            mProjOptions = "";

            try {
```

```
                  int remoteUserId =
ServletUtil.getRemoteUserID(mRemoteUser);
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = (TpteamUser) s.createQuery(
                              "from TpteamUser as user where user.id =
" + remoteUserId)
                              .uniqueResult();
                  projs = user.getProjects();
                  // Concatenate id with name to save database call
later.
                  for (Project proj : projs) {
                        mProjOptions += "<option value=\"" +
proj.getId() + ":"
                                    + proj.getName() + "\">" +
proj.getName() + "</option>";
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (!mProjOptions.equals(""))
                  mIsProjAvailable = true;

            mProjOptions = "<option selected>Choose Project</option>"
                        + mProjOptions;

            return mProjOptions;
      }
}
```

AddTestEntityByUser.java

```
/*********************************************************************
 *
 * File     :     AddTestEntityByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet used for adding new Tests to the TPTeam
 *                database by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.add;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.add.AddTestEntity;

public class AddTestEntityByUser extends AddTestEntity implements
UserServlet{
      private static final long serialVersionUID = 1L;

}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete

DeleteTest2ByUser.java

```
/**********************************************************************
 *
 * File     :     DeleteTest2ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Test by selecting it from the Test tree,
 *                for by a non-administrative Web user
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTest2;

public class DeleteTest2ByUser extends DeleteTest2 implements
UserServlet {
      private static final long serialVersionUID = 1L;
}
```

DeleteTestByUser.java

```
/*********************************************************************
 *
 * File     :     DeleteTestByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for deleting
 *                a TPTeam Test by first selecting its parent Project,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete;

import java.util.Set;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTest;

/*********************************************************************
 * File           :     DeleteTestByUser.java
 *
 * Description    :     Servlet that displays an input form for
 *                      deleting a TPTeam Test by first selecting its
 *                      parent Project, for use by a non-administrative
 *                      Web user
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class DeleteTestByUser extends DeleteTest implements UserServlet
{

      private static final long serialVersionUID = 1L;

      /**
       * Gets all TPTeam Projects that are associated
       * with the TPTeam user and wraps them
       * into HTML select option tags
       *
       * @return The Project select option tags
       * @throws Exception
       */
      protected String getProjRows() throws Exception {

            Transaction tx = null;
            Set<Project> projs = null;
            StringBuffer projRows = new StringBuffer();
            try {
```

```
                  int remoteUserId =
ServletUtil.getRemoteUserID(mRemoteUser);
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = (TpteamUser) s.createQuery(
                              "from TpteamUser as user where user.id =
" + remoteUserId)
                              .uniqueResult();
                  projs = user.getProjects();
                  for (Project proj : projs) {
                        String desc = proj.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId()
                                    + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;
            mProjRows = projRows.toString();
            return mProjRows;
      }

}
```

DeleteTestEntityByUser.java

```
/*********************************************************************
 *
 * File     :     DeleteTestEntityByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that deletes a Test from the TPTeam
 *                database, for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.delete;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.delete.DeleteTestEnt
ity;

public class DeleteTestEntityByUser extends DeleteTestEntity implements
UserServlet {
      private static final long serialVersionUID = 1L;
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec

ExecTest2ByUser.java

```
/*********************************************************************
 *
 * File     :     ExecTest2ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for executing
 *                a TPTeam Test by selecting it from the Test tree,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.ExecTest2;

public class ExecTest2ByUser extends ExecTest2 implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ExecTestByUser.java

```
/*********************************************************************
 *
 * File     :     ExecTestByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for executing
 *                a TPTeam Test by first selecting its parent Project,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.ExecTest;

public class ExecTestByUser extends ExecTest implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

UserProcessTestExec.java

```
/*********************************************************************
 *
 * File     :     UserProcessTestExec.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that validates a test request
 *                and delegates its execution, for use by a
 *                non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.exec;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.exec.AdminProcessTes
tExec;

public class UserProcessTestExec extends AdminProcessTestExec
implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update

UpdateTestByUser.java

```
/**********************************************************************
 *
 * File     :     UpdateTestByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Test by first selecting its parent Project,
 *                for use by a non-administrative Web user
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import java.util.Set
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest;

/**********************************************************************
 * File           :     UpdateTestByUser.java
 *
 * Description    :     Servlet that displays an input form for
 *                      updating a TPTeam Test by first selecting its
 *                      parent Project, for use by a non-administrative
 *                      Web user
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class UpdateTestByUser extends UpdateTest implements UserServlet
{
      private static final long serialVersionUID = 1L;

      /**
       * Gets all TPTeam Projects that the TPTeam
       * user is associated with and wraps them
       * into HTML select option tags
       *
       * @return The Project select option tags
       * @throws Exception
       */
      protected String getProjRows() throws Exception {
            Transaction tx = null;
            Set<Project> projs = null;
```

```
            StringBuffer projRows = new StringBuffer();
            try {
                  int remoteUserId =
ServletUtil.getRemoteUserID(mRemoteUser);
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  //Session s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = (TpteamUser) s.createQuery(
                              "from TpteamUser as user where user.id =
" + remoteUserId)
                              .uniqueResult();
                  projs = user.getProjects();
                  // Concatenate id with name to save database call
later.
                  for (Project proj : projs) {
                        String desc = proj.getDescription();
                        if (desc == null ||
desc.equalsIgnoreCase("null"))
                              desc = "";
                        projRows.append(rowNameHeader + proj.getName()
+ "</td>\n");
                        projRows.append(rowIDHeader + " value=\"" +
proj.getId()
                                    + "\">\n");
                        projRows.append(rowSubmitHeader);
                  }
                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if (projRows.length() > 0)
                  mIsProjAvailable = true;
            mProjRows = projRows.toString();

            return mProjRows;
      }
}
```

UpdateTestByUser2.java

```
/*********************************************************************
 *
 * File     :     UpdateTest2ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for updating
 *                a TPTeam Test by selecting it from the Test tree,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest2;

public class UpdateTestByUser2 extends UpdateTest2 implements
UserServlet {
      private static final long serialVersionUID = 1L;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateTest3\">\n";
}
```

UpdateTestByUser3.java

```
/*********************************************************************
 *
 * File     :     UpdateTestByUser3.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that loads the details for a particular Test
 *                into a form so that a user can edit and update them,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTest3;

public class UpdateTestByUser3 extends UpdateTest3 implements
UserServlet{
      private static final long serialVersionUID = 1L;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateTestEntity\">\n";
}
```

UpdateTestEntityByUser.java

```
/*********************************************************************
 *
 * File     :     UpdateTestEntityByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a Test in the TPTeam
 *                database, for use by a non-administrative
 *                Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateTestEnt
ity;

public class UpdateTestEntityByUser extends UpdateTestEntity implements
UserServlet{

      private static final long serialVersionUID = 1L;
}
```

UpdateUserProfile.java

```
/*********************************************************************
 *
 * File     :     UpdateUserProfile.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the details of the logged-in
 *                TPTeam user in a form for editable updating, for
 *                use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateUser2;

public class UpdateUserProfile extends UpdateUser2 implements
UserServlet{
      private static final long serialVersionUID = 1L;

      protected String mRemoteUser = null;

      protected String mFormAction = "<input type=\"hidden\"
name=\"formAction\" value=\"updateUserEntity\">\n";
}
```

UpdateUserProfileEntity.java

```
/*********************************************************************
 *
 * File     :     UpdateUserProfileEntity.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that updates a user in the TPTeam
 *                database, for use by a non-administrative
 *                Web user
 *
 *********************************************************************/

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.update;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.update.UpdateUserEnt
ity;

public class UpdateUserProfileEntity extends UpdateUserEntity
implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view

ViewProductByUser.java

```
/**********************************************************************
 *
 * File     :     ViewProductByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for selecting
 *                a particular TPTeam Product for viewing, for use
 *                by a non-administrative Web user
 *
 **********************************************************************/

package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProduct;

public class ViewProductByUser extends ViewProduct implements
UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ViewProject2ByUser.java

```
/*********************************************************************
 *
 * File     :     ViewProject2ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the view details
 *                a particular TPTeam Project, for use by a
 *                non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProject2;

public class ViewProject2ByUser extends ViewProject2 implements
UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ViewProjectByUser.java

```
/*********************************************************************
 *
 * File     :     ViewProjectByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a particular TPTeam Project, for use by a
 *                non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import java.util.Set;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Project;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.ServletUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewProject;

public class ViewProjectByUser extends ViewProject implements
UserServlet {
      private static final long serialVersionUID = 1L;

      protected String getProjRows() throws Exception
{
            Transaction tx = null;
            Set<Project> projs = null;
            StringBuffer projRows = new StringBuffer();
            try {
                  int remoteUserId =
ServletUtil.getRemoteUserID(mRemoteUser);
                  // For standalone
                  //Session s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  Session s =
Activator.getDefault().getHiberSessionFactory()
                              .getCurrentSession();
                  tx = s.beginTransaction();
                  TpteamUser user = (TpteamUser) s.createQuery(
                              "from TpteamUser as user where user.id =
" + remoteUserId)
                              .uniqueResult();
                  projs = user.getProjects();

                  for(Project proj : projs)
                  {
                        projRows.append(rowNameHeader +
proj.getName());
                        if(proj.getDescription() != null &&
!proj.getDescription().equalsIgnoreCase("null"))
```

```
                        {
                              projRows.append(": " +
proj.getDescription());
                        }
                        projRows.append("</td>\n" + rowIDHeader + "
value=\"" + proj.getId() + "\">\n" + rowSubmitHeader + "\n");
                  }

                  tx.commit();
            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
            if(projRows.length() > 0)
                  mIsProjAvailable = true;

            mProjRows = projRows.toString();

            return mProjRows;
      }
}
```

ViewTest2ByUser.java

```
/*********************************************************************
 *
 * File     :     ViewTest2ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a TPTeam Test by selecting it from the Test tree,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest2;

public class ViewTest2ByUser extends ViewTest2 implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ViewTest3ByUser.java

```
/*********************************************************************
 *
 * File     :     ViewTest3ByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that loads the details for a particular Test
 *                so that a user can view them, for use by a
 *                non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest3;

public class ViewTest3ByUser extends ViewTest3 implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ViewTestByUser.java

```
/*********************************************************************
 *
 * File     :     ViewTestByUser.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays an input form for viewing
 *                a TPTeam Test by first selecting its parent Project,
 *                for use by a non-administrative Web user
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewTest;

public class ViewTestByUser extends ViewTest implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

ViewUserProfile.java

```
/**********************************************************************
 *
 * File     :     ViewUserProfile.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Servlet that displays the details of the logged-in
 *                TPTeam user
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.http.user.view;

import edu.harvard.fas.rbrady.tpteam.tpmanager.http.UserServlet;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.http.admin.view.ViewUser2;

public class ViewUserProfile extends ViewUser2 implements UserServlet {
      private static final long serialVersionUID = 1L;
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.tpbridge

TPBridgeClient.java

```
/*****************************************************************
 *
 * File     :     TPBridgeClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Convenience client for the TPBridge OSGi Service
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tpbridge;

import java.util.ArrayList;
import org.eclipse.ecf.core.ContainerFactory;
import org.eclipse.ecf.core.IContainer;
import org.eclipse.ecf.core.identity.ID;
import org.eclipse.ecf.core.identity.IDFactory;
import org.eclipse.ecf.core.identity.Namespace;
import org.eclipse.ecf.core.security.ConnectContextFactory;
import org.eclipse.ecf.core.util.ECFException;
import org.osgi.framework.BundleContext;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.Client;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/*****************************************************************
 * File           :     TPBridgeClient.java
 *
 * Description    :     Convenience client for the TPBridge OSGi
 *                      Service
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*****************************************************************/
public class TPBridgeClient extends Client {

      protected static String CONTAINER_TYPE = "ecf.xmpps.smack";

      private Namespace mNamespace = null;

      private IContainer mContainer = null;

      private ID mTargetID = null;

      /**
       * Constructor,  initializes the XMPPS communication
       * container and gets a connection with the ID and password
       * given in the tpteam.properties file
       *
       * @param context the TPManager plug-in context
```

```
	 */
	public TPBridgeClient(BundleContext context) {
		super(context);
		try {
			setupContainer();
			// Then read tpteam.properties file and connect

	setTPMgrECFID(getTPTeamProps().getProperty(TPMANAGER_ECFID_KEY));
			String tpMgrPass = getTPTeamProps().getProperty(
					TPMANAGER_ECFID_PASSWORD);
			doConnect(getTPMgrECFID(), tpMgrPass);
			setContainer(getContainer(), mTargetID.getName(),
					ITPBridge.TPTEAM_MGR);

		} catch (ECFException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	/**
	 * Creates the ECF XMPPS Container and Namespace
	 * member variables
	 *
	 * @return the ECF XMPPS Container
	 * @throws ECFException
	 */
	protected IContainer setupContainer() throws ECFException {
		if (mContainer == null) {
			mContainer =
ContainerFactory.getDefault().createContainer(
					CONTAINER_TYPE);
			mNamespace = mContainer.getConnectNamespace();
		}
		return mContainer;
	}


	/**
	 * Initializes the ECF XMPPS Container and
	 * gets an XMPPS connection
	 *
	 * @param account the XMPPS account ID
	 * @param password the XMPPS plain text password
	 * @throws ECFException
	 */
	public void connect(String account, String password) throws
ECFException {
		setupContainer();
		doConnect(account, password);
	}

	/**
	 * Binds the ECF XMPPS Container with a connection
	 *
	 * @param account the XMPPS account ID
```

```
	 * @param password the XMPPS plain text password
	 * @throws ECFException
	 */
	protected void doConnect(String account, String password)
				throws ECFException {
		// Now connect
		ID targetID = IDFactory.getDefault().createID(mNamespace,
account);
		mContainer.connect(targetID, ConnectContextFactory
				.createPasswordConnectContext(password));
		mTargetID = targetID;
	}

	/**
	 * Determine if the ECF XMPPS Container is
	 * connected
	 *
	 * @return true if connected, false otherwise
	 */
	public synchronized boolean isConnected() {
		if (mContainer == null)
			return false;
		return (mContainer.getConnectedID() != null);
	}

	/**
	 * Close the ECF XMPPS Container connection
	 */
	public synchronized void close() {
		if (mContainer != null) {
			mContainer.dispose();
			mContainer = null;
			mTargetID = null;
		}
	}

	/**
	 * Gets the list of all TPEvents received out-of-JVM
	 * by the TPBridge OSGi Service
	 *
	 * @return the List of TPEvents
	 */
	public ArrayList<TPEvent> getEventLog() {
		ITPBridge tpBridge = (ITPBridge)
mServiceTracker.getService();
		if (tpBridge == null) {
			System.out
					.println("TPManager TPBridgeClient:
tpBridge service ref is null");
		}
		return tpBridge.getEventLog();
	}

	// Member variable accessors

	protected IContainer getContainer() {
		return mContainer;
```

```
	}

	protected Namespace getConnectNamespace() {
		return mNamespace;
	}
}
```

package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp

AutomationClient.java

```
/*********************************************************************
 *
 * File     :     AutomationClient.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A wrapper around the TPTP AutomationClientAdapter,
 *                utilizes the TPTP AgentController to execute TPTP
 *                test suites on behalf of the TPManager
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp;

import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import
org.eclipse.hyades.automation.client.adapters.java.AutomationClientAdap
ter;

/*********************************************************************
 * File           :     AutomationClient.java
 *
 * Description    :     A wrapper around the TPTP
 *                      AutomationClientAdapter,
 *                      utilizes the TPTP AgentController to execute
 *                      TPTP test suites on behalf of the TPManager
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class AutomationClient {

      /**
       * Runs the given TPTP test suites on the
       * machine configuration specified by input
       * paramters
       *
       * @param eclipse the Eclipse Home path
       * @param workspace the Eclipse Workspace path
       * @param project the Eclipse project inside the workspace
       * @param suites the TPTP test suites to be run
       * @param report the path to place any generated reports
       * @param connection the TPTP connection to the machine, local or
remote
       * @return the test execution verdict
       */
      @SuppressWarnings("unchecked")
      public String run(String eclipse, String workspace, String
project,
```

```
                  String[] suites, String report, String connection) {

            // Instantiate TPTP automatable services' Java automation
client adapter
            AutomationClientAdapter adapter = new
AutomationClientAdapter(eclipse);

            Properties properties = new Properties();
            properties.setProperty("connection", connection);
            properties.setProperty("quiet", "true");

            adapter

      .execute("org.eclipse.hyades.test.tools.core.verify",
                                    properties);

            boolean verified =
Boolean.valueOf(properties.getProperty("verified"))
                        .booleanValue();

            if (!verified) {
                  return "error";
            }

            System.out.println("AgentController is running...");

            final List results;

            properties = new Properties();
            properties.setProperty("workspace", workspace);
            properties.setProperty("project", project);

            // Create list of test suites for suite property
            int count = suites.length;
            List<String> list = new ArrayList<String>(count);
            for (int i = 0; i < count; i++) {
                  list.add(suites[i]);
            }

            // Add list to suite property
            properties.put("suite", list);

            // Execute test results interrogation service

      adapter.execute("org.eclipse.hyades.test.tools.core.execute",
                        properties);

            // Retrieve results back from return list
            results = (List) properties.get("results");
            System.out.println("Results collected: " + results);

            // Configure service properties for this service invocation
            properties = new Properties();
            properties.setProperty("workspace", workspace);
            properties.setProperty("project", project);
            properties.setProperty("quiet", "true");
```

```
            // Add list to results property
            properties.put("results", results);

            // Execute test results interrogation service

      adapter.execute("org.eclipse.hyades.test.tools.core.interrogate",
                        properties);

            // Output verdict indicated from test results
            System.out.println("Verdict: '" +
properties.getProperty("verdict")
                        + "'");

            return properties.getProperty("verdict");

      }
}
```

TPManager.java

```
/*********************************************************************
 *
 * File     :     TPManager.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Receives out-of-JVM TPEvent requests from remote
 *                TPBuddies or Web clients, opens new TPManagerThread
 *                to handle them
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp;

import java.util.Observable;
import java.util.Observer;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.eventadmin.EventAdminHandler;

/*********************************************************************
 * File           :     TPManager.java
 *
 * Description    :     Receives out-of-JVM TPEvent requests from
 *                      remote TPBuddies or Web clients, opens new
 *                      TPManagerThread  to handle them
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class TPManager implements Observer {

      /**
       * Receives notifications of out-of-JVM TPEvent requests or
       * Web client requests from the TPManager EventAdminHandler,
       * starts a new TPManagerThread to handle them.
       *
       * @see EventAdminHandler
       * @see TPManagerThread
       * @param observable the EventAmdinHandler object being observed
       * @param object the TPEvent object
       */
      public void update(Observable observable, Object object) {
            if (observable instanceof EventAdminHandler
                        && object instanceof TPEvent) {
                  TPEvent tpEvent = (TPEvent) object;
                  TPManagerThread tpManagerThread = new
TPManagerThread(tpEvent);
                  Thread tpThread = new Thread(tpManagerThread);
                  tpThread.start();
            }
      }
}
```

TPManagerThread.java

```
/**********************************************************************
 *
 * File     :     TPManagerThread.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The Thread that instantiated to handle each and
 *                every TPEvent request to the TPManager
 *
 **********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp;

import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;

/**********************************************************************
 * File           :     TPManagerThread.java
 *
 * Description    :     The Thread that instantiated to handle each and
 *                      every TPEvent request to the TPManager
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 **********************************************************************/
public class TPManagerThread implements Runnable {

      /** The TPEvent whose request is being handled */
      private TPEvent mTPEvent;

      /**
       * Constructor
       * @param tpEvent the TPEvent request
       */
      public TPManagerThread(TPEvent tpEvent) {
            mTPEvent = tpEvent;
      }

      /**
       * Performs the necessary CRUD or execution operation
       * based upon the TPEvent topic.  Sends a response when
       * appropriate.
       */
      public void run() {
            String tpTopic = mTPEvent.getTopic();
            try {
                  System.out.println("NEW THREAD w/Topic: " + tpTopic);
                  if (tpTopic.equals(ITPBridge.PROJ_GET_REQ_TOPIC)) {
                        TPTestCRUD.sendProjGetResponse(mTPEvent);
                  } else if
(tpTopic.equalsIgnoreCase(ITPBridge.TEST_EXEC_REQ_TOPIC)) {
                        TPTestExec.runTest(mTPEvent.getID(), mTPEvent);
                        TPTestExec.sendTestExecResponse(mTPEvent);
                  } else if (tpTopic
```

```
      .equalsIgnoreCase(ITPBridge.TEST_TREE_GET_REQ_TOPIC)) {
                        TPTestCRUD.sendTestTreeGetResponse(mTPEvent);
                  } else if
(tpTopic.equalsIgnoreCase(ITPBridge.TEST_DEL_REQ_TOPIC)) {
                        TPTestCRUD.deleteTest(mTPEvent);
                        TPTestCRUD.sendDelTestResponse(mTPEvent);
                  } else if (tpTopic

      .equalsIgnoreCase(ITPBridge.TEST_DETAIL_REQ_TOPIC)) {
                        TPTestCRUD.sendTestDetailResponse(mTPEvent);
                  } else if (tpTopic

      .equalsIgnoreCase(ITPBridge.TEST_UPDATE_DATA_REQ_TOPIC)) {
                        TPTestCRUD.sendTestUpdateDatResponse(mTPEvent);
                  } else if (tpTopic

      .equalsIgnoreCase(ITPBridge.TEST_UPDATE_REQ_TOPIC)) {
                        TPTestCRUD.sendTestUpdateResponse(mTPEvent);
                  } else if
(tpTopic.equalsIgnoreCase(ITPBridge.TEST_ADD_REQ_TOPIC)) {
                        TPTestCRUD.sendTestAddResponse(mTPEvent);
                  } else if (tpTopic

      .equalsIgnoreCase(ITPBridge.CHART_GET_DATA_REQ_TOPIC)) {
                        TPTestCRUD.sendChartDataResponse(mTPEvent);
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

TPTestCRUD.java

```
/*****************************************************************
 *
 * File     :     TPTestCRUD.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides synchronized methods for performing
 *                TPTeam CRUD operations and sending the resulting
 *                responses
 *
 *****************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp;

import java.util.Hashtable;
import java.util.Set;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.chart.ChartDataSet;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate.ChartUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate.ProjectUtil;
import edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate.TestUtil;

/*****************************************************************
 * File           :     TPTestCRUD.java
 *
 * Description    :     Provides synchronized methods for performing
 *                      TPTeam CRUD operations and sending the
 *                      resulting responses
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady

*****************************************************************/
public class TPTestCRUD {

      /**
       * Sends a TPEvent response when a new Test folder or definition
has been
       * persisted to the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent add Test request
       * @throws Exception
       */
      public static synchronized void sendTestAddResponse(TPEvent
tpEvent)
                  throws Exception {
```

```
            TestUtil.addTest(tpEvent);

            Hashtable<String, String> dictionary =
tpEvent.getDictionary();
            dictionary.put(TPEvent.SEND_TO,
dictionary.get(TPEvent.FROM));
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());

            System.out.println("TPManager.sendTestAddResponse: Send To:
"
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
                        + dictionary.get(TPEvent.FROM));

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_ADD_RESP_TOPIC, dictionary);

      }

      /**
       * Sends a TPEvent response when the Test folder or definition
details have
       * been lodaded from the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent Test details request
       * @throws Exception
       */
      public static void sendTestDetailResponse(TPEvent tpEvent) throws
Exception {
            Hashtable<String, String> dictionary =
tpEvent.getDictionary();
            dictionary.put(TPEvent.SEND_TO,
dictionary.get(TPEvent.FROM));
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());

            System.out.println("TPManager.sendTestDetailResponse: Send
To: "
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
                        + dictionary.get(TPEvent.FROM));

            Test test = TestUtil.getTestByID(tpEvent.getID(), true);
            dictionary.put(TPEvent.TEST_PROP_XML_KEY,
TestXML.getTestPropXML(test));

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_DETAIL_RESP_TOPIC, dictionary);
      }

      /**
       * Sends a TPEvent response when the Test folder or definiton
details for an
       * update have been loaded from the TPTeam database
```

```
       *
       * @param tpEvent
       *            the incoming TPEvent Test update details request
       * @throws Exception
       */
      public static void sendTestUpdateDatResponse(TPEvent tpEvent)
                  throws Exception {
            Hashtable<String, String> dictionary =
tpEvent.getDictionary();
            dictionary.put(TPEvent.SEND_TO,
dictionary.get(TPEvent.FROM));
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());

            System.out.println("TPManager.sendTestUpdateDatResponse:
Send To: "
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
                        + dictionary.get(TPEvent.FROM));

            Test test = TestUtil.getTestByID(tpEvent.getID(), false);
            Test testStub = TestUtil.getTestUpdateStub(test);

            dictionary.put(TPEvent.TEST_XML_KEY,
TestXML.getXML(testStub));

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_UPDATE_DATA_RESP_TOPIC,
dictionary);
      }

      /**
       * Sends a TPEvent response when the Test folder or definiton
details for an
       * update have been persisted to the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent Test update request
       * @throws Exception
       */
      public static synchronized void sendTestUpdateResponse(TPEvent
tpEvent)
                  throws Exception {

            TestUtil.updateTest(tpEvent);

            Hashtable<String, String> dictionary =
tpEvent.getDictionary();
            dictionary.put(TPEvent.SEND_TO,
dictionary.get(TPEvent.FROM));
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());

            System.out.println("TPManager.sendTestUpdateResponse: Send
To: "
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
```

```
                        + dictionary.get(TPEvent.FROM));

            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_UPDATE_RESP_TOPIC, dictionary);

      }

      /**
       * Deletes a Test folder or definition from the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent Test delete request
       * @throws Exception
       */
      public static synchronized void deleteTest(TPEvent tpEvent)
                  throws Exception {
            Transaction tx = null;

            try {
                  // If running in OSGi runtime, get Session from
plugin
                  Session s = null;
                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();

                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }

                  tx = s.beginTransaction();
                  Test test = (Test) s.load(Test.class, new
Integer(tpEvent.getID()));

                  // Collect project member ecfIDs
                  Set<TpteamUser> users =
test.getProject().getTpteamUsers();
                  StringBuilder userECFIDs = new StringBuilder();
                  int idx = 0;
                  for (TpteamUser user : users) {
                        if (idx == 0)
                              userECFIDs.append(user.getEcfId());
                        else
                              userECFIDs.append("/" + user.getEcfId());
                        idx++;
                  }
                  String ECFID =
tpEvent.getDictionary().get(TPEvent.FROM);
                  tpEvent.getDictionary().put(TPEvent.ECFID_KEY,
ECFID);
                  tpEvent.getDictionary().put(TPEvent.FROM,
userECFIDs.toString());
                  tpEvent.getDictionary().put(TPEvent.TEST_NAME_KEY,
test.getName());
```

```
                s.delete(test);
                s.flush();
                tx.commit();
          } catch (Exception e) {
                if (tx != null)
                      tx.rollback();
                throw e;
          }
    }

    /**
     * Sends a TPEvent response when the Test folder or definition
have been
     * deleted from the TPTeam database
     *
     * @param tpEvent
     *            the incoming TPEvent Test delete request
     * @throws Exception
     */
    public static void sendDelTestResponse(TPEvent tpEvent) throws
Exception {
          Hashtable<String, String> dictionary =
tpEvent.getDictionary();
          dictionary.put(TPEvent.SEND_TO,
tpEvent.getDictionary().get(
                      TPEvent.FROM));
          dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                      .getTPMgrECFID());
          dictionary.put(TPEvent.ID_KEY, tpEvent.getID());
          System.out.println("TPManager.sendTestDelResponse: Send To:
"
                      + dictionary.get(TPEvent.SEND_TO) + ", From: "
                      + dictionary.get(TPEvent.FROM));
          Activator.getDefault().getEventAdminClient().sendEvent(
                      ITPBridge.TEST_DEL_RESP_TOPIC, dictionary);
    }

    /**
     * Sends a TPEvent response when the Test Project details have
been loaded
     * from the TPTeam database
     *
     * @param tpEvent
     *            the incoming TPEvent get project request
     * @throws Exception
     */
    public static void sendProjGetResponse(TPEvent tpEvent) throws
Exception {
          Hashtable<String, String> dictionary = new
Hashtable<String, String>();
          dictionary.put(TPEvent.SEND_TO,
tpEvent.getDictionary().get(
                      TPEvent.FROM));
          dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                      .getTPMgrECFID());
```

```
            dictionary.put(TPEvent.PROJ_PROD_XML_KEY, ProjectUtil
                        .getProjProdXML(tpEvent));
            System.out.println("TPManager.sendProjGetResponse: Send To:
"
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
                        + dictionary.get(TPEvent.FROM));
            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.PROJ_GET_RESP_TOPIC, dictionary);
      }

      /**
       * Sends a TPEvent response when the Test Tree details have been
loaded from
       * the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent get test tree request
       * @throws Exception
       */
      public static void sendTestTreeGetResponse(TPEvent tpEvent)
                  throws Exception {
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO,
tpEvent.getDictionary().get(
                        TPEvent.FROM));
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());
            dictionary.put(TPEvent.TEST_TREE_XML_KEY, TestUtil
                        .getTestTreeXML(tpEvent));
            System.out.println("TPManager.sendTestTreeGetResponse: Send
To: "
                        + dictionary.get(TPEvent.SEND_TO) + ", From: "
                        + dictionary.get(TPEvent.FROM));
            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.TEST_TREE_GET_RESP_TOPIC,
dictionary);
      }

      /**
       * Sends a TPEvent response when the TPTeam Chart details have
been loaded
       * from the TPTeam database
       *
       * @param tpEvent
       *            the incoming TPEvent get chart data request
       * @throws Exception
       */
      public static void sendChartDataResponse(TPEvent tpEvent) throws
Exception {
            Hashtable<String, String> dictionary = new
Hashtable<String, String>();
            dictionary.put(TPEvent.SEND_TO,
tpEvent.getDictionary().get(
                        TPEvent.FROM));
```

```
            dictionary.put(TPEvent.FROM,
Activator.getDefault().getTPBridgeClient()
                        .getTPMgrECFID());
            dictionary.put(ChartDataSet.CHART_TYPE,
tpEvent.getDictionary().get(
                        ChartDataSet.CHART_TYPE));

            if
(tpEvent.getDictionary().get(ChartDataSet.CHART_TYPE).equals(
                        ChartDataSet.PIE)) {
                  dictionary.put(TPEvent.CHART_DATASET_XML_KEY,
ChartUtil
                              .getPieChartXML(tpEvent));
            } else if
(tpEvent.getDictionary().get(ChartDataSet.CHART_TYPE).equals(
                        ChartDataSet.BAR)) {
                  dictionary.put(TPEvent.CHART_DATASET_XML_KEY,
ChartUtil
                              .getBarChartXML(tpEvent));
            } else if
(tpEvent.getDictionary().get(ChartDataSet.CHART_TYPE).equals(
                        ChartDataSet.LINE)) {
                  dictionary.put(TPEvent.CHART_DATASET_XML_KEY,
ChartUtil
                              .getLineChartXML(tpEvent));
            }
            Activator.getDefault().getEventAdminClient().sendEvent(
                        ITPBridge.CHART_GET_DATA_RESP_TOPIC,
dictionary);
      }
}
```

TPTestExec.java

```
/*********************************************************************
 *
 * File     :     TPTestExec.java
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides synchronized methods for performing
 *                TPTeam Test executions and sending the resulting
 *                responses
 *
 *********************************************************************/
package edu.harvard.fas.rbrady.tpteam.tpmanager.tptp;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.Set;
import org.hibernate.Session;
import org.hibernate.Transaction;
import edu.harvard.fas.rbrady.tpteam.tpbridge.bridge.ITPBridge;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.HibernateUtil;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.JunitTest;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.Test;
import edu.harvard.fas.rbrady.tpteam.tpbridge.hibernate.TpteamUser;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEntity;
import edu.harvard.fas.rbrady.tpteam.tpbridge.model.TPEvent;
import edu.harvard.fas.rbrady.tpteam.tpbridge.xml.TestExecutionXML;
import edu.harvard.fas.rbrady.tpteam.tpmanager.Activator;
import
edu.harvard.fas.rbrady.tpteam.tpmanager.hibernate.TestExecutionUtil;

/*********************************************************************
 * File           :     TPTestExec.java
 *
 * Description    :     Provides synchronized methods for performing
 *                      TPTeam Test executions and sending the
 *                      resulting responses
 *
 * @author Bob Brady, rpbrady@gmail.com
 * @version $Revision$
 * @date $Date$ Copyright (c) 2007 Bob Brady
 *********************************************************************/
public class TPTestExec {

      /** Locale to format test result time */
      public static final Locale USA_LOCALE = new Locale("en", "US");

      /** DateTime format for test reusult */
      public static final SimpleDateFormat SIMPLE_DATE_FORMATTER = new
SimpleDateFormat(
                  "MM/dd/yyyy HH:mm:ss.SSS z", USA_LOCALE);

      /**
       * Loads the test metadata from the TPTeam database
       * for a given Test and then executes the Test
```

```
       *
       * @param testID the TPTeam ID of the Test
       * @param tpEvent the TPEvent get test execution request
       * @throws Exception
       */
      public static synchronized void runTest(String testID, TPEvent
tpEvent)
                  throws Exception {
            Transaction tx = null;
            String testType = null;

            try {
                  // If running in OSGi runtime, get Session from
plugin
                  Session s = null;
                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();

                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }
                  tx = s.beginTransaction();

                  Test test = (Test) s.load(Test.class, new
Integer(testID));
                  testType = test.getTestType().getName();

                  // Collect project member ecfIDs
                  Set<TpteamUser> users =
test.getProject().getTpteamUsers();
                  StringBuilder userECFIDs = new StringBuilder();
                  int idx = 0;
                  for (TpteamUser user : users) {
                        if (idx == 0)
                              userECFIDs.append(user.getEcfId());
                        else
                              userECFIDs.append("/" + user.getEcfId());
                        idx++;
                  }
                  String ECFID =
tpEvent.getDictionary().get(TPEvent.FROM);
                  tpEvent.getDictionary().put(TPEvent.ECFID_KEY,
ECFID);
                  tpEvent.getDictionary().put(TPEvent.SEND_TO,
userECFIDs.toString());
                  tpEvent.getDictionary().put(TPEvent.TEST_NAME_KEY,
test.getName());

                  s.flush();
                  tx.commit();

                  if (testType.equalsIgnoreCase("JUNIT")) {
                        runJUnitTest(testID, tpEvent);
                  }
```

```
                  TestExecutionUtil.insertTestExec(testID, tpEvent);

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }

      }

      /**
       * Helper method that collects the JUnit metadata for
       * a given test and requests the AutomationClient
       * execute it
       *
       * @see AutomationClient
       * @param testID the TPTeam ID of the Test
       * @param tpEvent the TPEvent get test execution request
       * @throws Exception
       */
      private static synchronized void runJUnitTest(String testID,
TPEvent tpEvent)
                  throws Exception {
            Transaction tx = null;
            String eclipseHome = null;
            String workspace = null;
            String project = null;
            String suite = null;
            String tptpConn = null;
            String report = null;

            try {
                  // If running in OSGi runtime, get Session from
plugin
                  Session s = null;
                  if (Activator.getDefault() != null) {
                        s =
Activator.getDefault().getHiberSessionFactory()
                                    .getCurrentSession();

                  } else {
                        s =
HibernateUtil.getSessionFactory().getCurrentSession();
                  }
                  tx = s.beginTransaction();

                  Test test = (Test) s.load(Test.class, new
Integer(testID));
                  JunitTest junit = test.getJunitTests().toArray(new
JunitTest[0])[0];

                  eclipseHome = junit.getEclipseHome();
                  workspace = junit.getWorkspace();
                  project = junit.getProject();
                  suite = junit.getTestSuite();
                  report = junit.getReportDir();
                  tptpConn = junit.getTptpConnection();
```

```
                  s.flush();
                  tx.commit();

                  AutomationClient automationClient = new
AutomationClient();
                  String verdict = automationClient.run(eclipseHome,
workspace,
                              project, new String[] { suite }, report,
tptpConn);

                  System.out.println("Test Verdict: " + verdict);

                  tpEvent.getDictionary().put(TPEvent.VERDICT_KEY,
verdict);
                  tpEvent.setStatus(verdict);
                  tpEvent.getDictionary().put(TPEvent.TIMESTAMP_KEY,
getDateTime());

            } catch (Exception e) {
                  if (tx != null)
                        tx.rollback();
                  throw e;
            }
      }

      /**
       * Sends a Test execution response TPEvent
       * when the requested Test execution has
       * completed
       *
       * @param tpEvent the TPEvent Test execution request
       */
      public static void sendTestExecResponse(TPEvent tpEvent) {
            tpEvent.setTopic(ITPBridge.TEST_EXEC_RESULT_TOPIC);
            tpEvent.getDictionary().put(TPEvent.FROM,

      Activator.getDefault().getTPBridgeClient().getTPMgrECFID());

            TPEntity execEntity = TestExecutionXML
                        .getTPEntityFromExecEvent(tpEvent);
            String execXML =
TestExecutionXML.getTPEntityXML(execEntity);
            tpEvent.getDictionary().put(TPEvent.TEST_EXEC_XML_KEY,
execXML);

            System.out.println("TPManager runTests(): sending tpEvent
w/verdict "
                        +
tpEvent.getDictionary().get(TPEvent.VERDICT_KEY)
                        + " & topic " + tpEvent.getTopic() + " to "
                        +
tpEvent.getDictionary().get(TPEvent.SEND_TO));

            Activator.getDefault().getEventAdminClient().sendEvent(
                        tpEvent.getTopic(), tpEvent.getDictionary());
      }
```

```
/**
 * Helper method to get properly
 * formatted test execution timestamp
 *
 * @return the timestamp
 */
private static String getDateTime() {
      Date now = new Date();
      return SIMPLE_DATE_FORMATTER.format(now);
}
```

JavaScript Code

This section contains the JavaScript code that was used in TPTeam for the validation and confirmation of user input data. The files are listed in alphabetical order by name.

add_test_folder.js

```
/**********************************************************************
 *
 * File     :     add_test_folder.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used when adding test
 *                      folders
 *
 **********************************************************************/
function validatePage2Form ()
{
      if(selectedID == "")
      {
            alert("Please select a parent folder.");
            return false;
      }
      document.addTestPage2.parentID.value = selectedID;
      document.addTestPage2.action = "addTestEntity";
      document.addTestPage2.submit();
      return true;
}
```

add_test_tree.js

```
/*********************************************************************
 *
 * File     :     add_test_tree.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A script used to render a test tree when adding tests
 *
 *********************************************************************/
var openImg = new Image();
openImg.src = "/bridge/tpteam/images/open.gif";
var closedImg = new Image();
closedImg.src = "/bridge/tpteam/images/closed.gif";
var selectedID = "";

function showBranch(branch){
      var objBranch = document.getElementById(branch).style;
      if(objBranch.display=="block")
            objBranch.display="none";
      else
      {
            objBranch.display="block";
      }
}

function swapFolder(img){
      objImg = document.getElementById(img);
      if(objImg.src.indexOf('closed.gif')>-1)
            objImg.src = openImg.src;
      else
            objImg.src = closedImg.src;
}

function makeBold(id){
      if(selectedID != "")
      {
            document.getElementById(selectedID).style.fontWeight =
'normal';
      }
      document.getElementById(id).style.fontWeight = 'bold';
      selectedID = id;
}
```

add_test_type.js

```
/*********************************************************************
 *
 * File     :     add_test_type.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used when adding test
 *                      types
 *
 *********************************************************************/
function validatePage2Form ()
{
      var workspace = document.addTestPage2.eclipseWorkspace.value;
      var home = document.addTestPage2.eclipseHome.value;
      var proj = document.addTestPage2.eclipseProj.value;
      var dir = document.addTestPage2.reportDir.value;
      var conn = document.addTestPage2.tptpConn.value;
      var suite = document.addTestPage2.testSuite.value;
      document.addTestPage2.parentID.value = selectedID;

      if( workspace == "")
      {
            alert("Please enter the eclipse workspace.");
            return false;
      }
      else if( home == "")
      {
            alert("Please enter the home directory of the eclipse
installation.");
            return false;
      }
      else if(proj == "")
      {
            alert("Please enter the name of the eclipse project
containing the test.");
            return false;
      }
      else if(dir == "")
      {
            alert("Please enter the directory where tptp results will
be stored.");
            return false;
      }
      else if(conn == "")
      {
            alert("Please enter the TPTP connection URL.");
            return false;
      }
      else if(suite == "")
      {
            alert("Please enter the test suite name.");
            return false;
      }
      else if(document.addTestPage2.parentID.value == "")
```

```
	{
		alert("Please select a parent folder.");
		return false;
	}
	document.addTestPage2.action = "addTestEntity";
	document.addTestPage2.submit();
	return true;
}
```

add_test.js

```
/*********************************************************************
 *
 * File     :     add_test.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used when adding tests,
 *                to verify core test data
 *
 *********************************************************************/
function validatePage1Form ()
{
      var name = document.addTestPage1.testName.value;
      var desc = document.addTestPage1.testDesc.value;
      var proj = document.addTestPage1.proj;
      var testType = document.addTestPage1.testType;

      if( name == "")
      {
            alert("Please enter a test name.");
            return false;
      }
      else if(proj.selectedIndex == 0)
      {
            alert("Please select a project.");
            return false;
      }
      else if(testType.selectedIndex == 0)
      {
            alert("Please select a testType.");
            return false;
      }
      document.addTestPage1.action = "addTest";
      document.addTestPage1.submit();
      return true;
}
```

add_user.js

```
/*********************************************************************
 *
 * File     :     add_user.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used when adding new TPTeam
 *                users
 *
 *********************************************************************/
function validateForm ()
{
      var firstName = document.addUser.firstName.value;
      var lastName = document.addUser.lastName.value;
      var userName = document.addUser.userName.value;
      var password = document.addUser.password.value;
      var email = document.addUser.email.value;
      var phone = document.addUser.phone.value;
      var ecfID = document.addUser.ecfID.value;
      var roleList = document.addUser.role;
      var projList = document.addUser.projects;

      if( firstName == "")
      {
            alert("Please enter a first name.");
            return false;
      }
      else if( lastName == "")
      {
            alert("Please enter a last name.");
            return false;
      }
      else if(userName == "")
      {
            alert("Please enter a userName.");
            return false;
      }
      else if(password == "")
      {
            alert("Please enter a password.");
            return false;
      }
      else if(phone == "" || !isPhoneValid(phone))
      {
            alert("Please enter a phone number in ddd-ddd-dddd
format.");
            return false;
      }
      else if(email == "" || !isEmailValid(email))
      {
            alert("Please enter an email address.");
            return false;
      }
      else if(ecfID == "" || !isEmailValid(ecfID))
```

```
	{
		alert("Please enter a valid ECF ID.");
		return false;
	}
	else if(roleList.selectedIndex == 0)
	{
		alert("Please select a role.");
		return false;
	}
	else if(projList.selectedIndex == 0)
	{
		alert("Please select one or more projects.");
		return false;
	}

	document.addUser.action = "addUserEntity";
	document.addUser.submit();
	return true;
}

function isEmailValid(email)
{
     var emailRegEx = "^[\\w-_\.]*[\\w-_\.]\@[\\w]\.+[\\w]+[\\w]$";
     var regex = new RegExp(emailRegEx);
     return regex.test(email);
  }

  function isPhoneValid(phone)
  {
     var phoneRegEx = "^\\d{3}-\\d{3}-\\d{4}$";
     var regex = new RegExp(phoneRegEx);
     return regex.test(phone);
  }
```

delete_prod.js

```
/*********************************************************************
 *
 * File     :     delete_prod.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form confirmation script used before the deleting
 *                      a Product
 *
 *********************************************************************/
      function validateForm (form)
      {
            var name = form.prodName.value;
            form.action = "deleteProductEntity";
            if(confirm("Do you really want to delete product \"" + name
+ "\"?"))
            {
                  form.submit();
            return true;
            }
            else
            {
                  return false;
            }
      }
```

delete_proj.js

```
/**********************************************************************
 *
 * File     :     delete_proj.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form confirmation script used before the deleting
 *                a Project
 *
 **********************************************************************/
      function validateForm (form)
      {
            form.action = "deleteProjectEntity";
            if(confirm("Do you really want to delete the project and
its entire test tree?"))
            {
                  form.submit();
            return true;
            }
            else
            {
                  return false;
            }
      }
```

delete_test_tree.js

```
/**********************************************************************
 *
 * File     :     delete_test_tree.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script to ensure a Test Tree
 *                element was selected before a deletion request
 *
 **********************************************************************/
function validateForm(form)
{
      if(selectedID == "")
      {
            alert("Please select a folder or test.");
            return false;
      }
      form.testID.value = selectedID;
      form.action = "deleteTestEntity";
      form.submit();
      return true;
}
```

delete_user.js

```
/**********************************************************************
 *
 * File     :     delete_user.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form confirmation script used before the deleting
 *                      a TPTeam user
 *
 **********************************************************************/
      function validateForm (form)
      {
            form.action = "deleteUserEntity";
            if(confirm("Do you really want to delete this user?"))
            {
                  form.submit();
            return true;
            }
            else
            {
                  return false;
            }
      }
```

exec_test_tree.js

```
/**********************************************************************
 *
 * File     :     exec_test_tree.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script to ensure a Test Tree
 *                element was selected before an execution request
 *
 **********************************************************************/
function validateForm(form)
{
      if(selectedID == "")
      {
            alert("Please select a folder or test.");
            return false;
      }
      if(selectedID == "0")
      {
            alert("Please select a folder or test under the root
node.");
            return false;
      }

      form.testID.value = selectedID;
      form.action = "processTestExec";
      form.submit();
      return true;
}
```

update_prod.js

```
/**********************************************************************
 *
 * File     :     update_prod.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used before the updating
 *                a Product
 *
 **********************************************************************/
      function validateForm (form)
      {
            var name = form.prodName.value;
            var desc = form.prodDesc.value;
            if( name == "")
            {
                  alert("Please enter at least a product name.");
                  return false;
            }
            form.action = "updateProductEntity";
            form.submit();
        return true;
      }
```

update_proj.js

```
/**********************************************************************
 *
 * File     :     update_proj.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used before the updating
 *                a Project
 *
 **********************************************************************/
function validateForm (form)
{
      var name = form.projName.value;
      var desc = form.projDesc.value;
      var prod = form.prod;
      if( name == "")
      {
            alert("Please enter at least a project name.");
            return false;
      }
      else if(prod.options[prod.selectedIndex].value == "")
      {
            alert("Please select a product.");
            return false;
      }
      form.action = "updateProjectEntity";
      form.submit();
      return true;
}
```

update_test_def.js

```
/*********************************************************************
 *
 * File     :     update_test_def.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used before updating
 *                a Test definition
 *
 *********************************************************************/
function validateForm(form)
{
      var name = form.testName.value;
      var workspace = form.eclipseWorkspace.value;
      var home = form.eclipseHome.value;
      var proj = form.eclipseProj.value;
      var dir = form.reportDir.value;
      var conn = form.tptpConn.value;
      var suite = form.testSuite.value;

      if( name == "")
      {
            alert("Please enter a name for the test element.");
            return false;
      }
      else if( workspace == "")
      {
            alert("Please enter the eclipse workspace.");
            return false;
      }
      else if( home == "")
      {
            alert("Please enter the home directory of the eclipse
installation.");
            return false;
      }
      else if(proj == "")
      {
            alert("Please enter the name of the eclipse project
containing the test.");
            return false;
      }
      else if(dir == "")
      {
            alert("Please enter the directory where tptp results will
be stored.");
            return false;
      }
      else if(conn == "")
      {
            alert("Please enter the TPTP connection URL.");
            return false;
      }
      else if(suite == "")
```

```
	{
		alert("Please enter the test suite name.");
		return false;
	}
	form.action = form.formAction.value;
	form.submit();
	return true;
}
```

update_test_folder.js

```
/**********************************************************************
 *
 * File     :     update_test_folder.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used before updating
 *                a Test folder
 *
 **********************************************************************/
function validateForm(form)
{
      var name = form.testName.value;
      if( name == "")
      {
            alert("Please enter the test element name.");
            return false;
      }
      form.action = form.formAction.value;
      form.submit();
      return true;
}
```

update_test_tree.js

```
/*********************************************************************
 *
 * File     :     update_test_tree.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script to ensure a Test Tree node
 *                was selected before a Test update is done
 *
 *********************************************************************/
function validateForm(form)
{
      if(selectedID == "")
      {
            alert("Please select a folder or test.");
            return false;
      }
      form.testID.value = selectedID;
      form.action = form.formAction.value;
      form.submit();
      return true;
}
```

update_user.js

```
/*********************************************************************
 *
 * File     :     update_user.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script used before updating
 *                a TPTeam user
 *
 *********************************************************************/
function validateForm (form)
{
      var firstName = form.firstName.value;
      var lastName = form.lastName.value;
      var userName = form.userName.value;
      var password = form.password.value;
      var passwordConfirm = form.passwordConfirm.value;
      var email = form.email.value;
      var phone = form.phone.value;
      var ecfID = form.ecfID.value;

      if( firstName == "")
      {
            alert("Please enter a first name.");
            return false;
      }
      else if( lastName == "")
      {
            alert("Please enter a last name.");
            return false;
      }
      else if(userName == "")
      {
            alert("Please enter a userName.");
            return false;
      }
      else if(password == "" || passwordConfirm == "")
      {
            alert("Please enter a password.");
            return false;
      }
      else if(password != passwordConfirm)
      {
            alert("Password not confirmed.  Please re-enter
passwords.");
            return false;
      }
      else if(phone == "" || !isPhoneValid(phone))
      {
            alert("Please enter a phone number in ddd-ddd-dddd
format.");
            return false;
      }
      else if(email == "" || !isEmailValid(email))
```

```
      {
            alert("Please enter an email address.");
            return false;
      }
      else if(ecfID == "" || !isEmailValid(ecfID))
      {
            alert("Please enter a valid ECF ID.");
            return false;
      }

      form.action = form.formAction.value;
      form.submit();
      return true;
}

function isEmailValid(email)
{
     var emailRegEx = "^[\\w-_\.]*[\\w-_\.]\@[\\w]\.+[\\w]+[\\w]$";
     var regex = new RegExp(emailRegEx);
     return regex.test(email);
  }

  function isPhoneValid(phone)
  {
     var phoneRegEx = "^\\d{3}-\\d{3}-\\d{4}$";
     var regex = new RegExp(phoneRegEx);
     return regex.test(phone);
  }
```

view_test_tree.js

```
/**********************************************************************
 *
 * File     :     view_test_tree.js
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A form validation script to ensure a Test Tree
 *                element was selected before a view request
 *
 **********************************************************************/
function validateForm(form)
{
      if(selectedID == "")
      {
            alert("Please select a folder or test.");
            return false;
      }
      if(selectedID == "0")
      {
            alert("Please select a folder or test under the root
node.");
            return false;
      }

      form.testID.value = selectedID;
      form.action = "viewTest3";
      form.submit();
      return true;
}
```

HTML Files

This section contains the HTML files that were part of the TPManager Web application. All files are listed in alphabetical order by name: first by the parent directory name, then by file name.

admin/index.html

```
<!--
**********************************************************************
 *
 * File     :     index.html
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The home Web page for administrative users
 *

**********************************************************************
-->
<html>
<head>
<title>TPManager Admin Home Page</title>
<script language="JavaScript">
<!--
      function validateForm ( )
      {
            var op = document.adminOp.op.value;
            var type = document.adminOp.type.value;
            if(op == "Choose" || type == "Choose")
            {
                  alert("Please select an operation and type to
perform");
                  return false;
            }
            var opType = op + type;
            if(opType == "addProduct")
                  opType += ".html";
            document.adminOp.action = op + "/" + opType;
            document.adminOp.submit();
        return true;
}
-->
</script>

</head>
<body>

<h3 align="center">TPManager Admin Home Page</h3>
<hr size="2">
```

```
<p>
<form name="adminOp" method="post" onSubmit="return validateForm( );">
<P>
<div align="center"><SELECT name="op">
      <OPTION selected value="Choose">Choose Op</OPTION>
      <OPTION value="add">Add</OPTION>
      <OPTION value="update">Update</OPTION>
      <OPTION value="delete">Delete</OPTION>
    <OPTION value="view">View</OPTION>
        <OPTION value="exec">Execute</OPTION>
</SELECT>     <SELECT name="type">
      <OPTION selected value="Choose">Choose Type</OPTION>
      <OPTION value="Product">Product</OPTION>
      <OPTION value="Project">Project</OPTION>
      <OPTION value="Test">Test</OPTION>
      <OPTION value="User">User</OPTION>
</SELECT>
<P><INPUT type="submit" value="Go"></P>
</FORM>
<hr size="2">
</body>
</html>
```

admin/add/addProduct.html

```
<!--
************************************************************************
 *
 * File     :     addProduct.html
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     Provides a form for adding a TPTeam Product via the
                  administrative Web user pages
 *
 ************************************************************************
-->
<html>
<head>
<title>TPManager Admin Home Page</title>
<script language="JavaScript">
<!--
      function validateForm ( )
      {
            var name = document.addProd.prodName.value;
            var desc = document.addProd.prodDesc.value;
            if( name == "")
            {
                  alert("Please enter at least a product name.");
                  return false;
            }
            document.addProd.action = "addProductEntity";
            document.addProd.submit();
        return true;
}
-->
</script>

</head>
<body>

<h3 align="center">TPManager Admin</h3>
<div align="center">
<a href="../index.html">Home</a>
<hr size="2">
<p>

<h4>Add Product</h4>
<form name="addProd" method="post" onSubmit="return validateForm( );">
<table >
<tr><th>Name:</th><td><input type="text" name="prodName"
size="25"></td></tr>
<tr><th>Description:</th><td><input type="text" name="prodDesc"
size="50"></td></tr>
</table>
<br>
<input type="submit" value="Add">
</form>
<p>
```

```
</div>
<hr size="2>
</body>
</html>
```

user/index.html

```
<!--
*************************************************************************
 *
 * File     :     index.html
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     The home Web page for non-administrative users
 *
 ************************************************************************-
->
<html>
<head>
<title>TPManager User Home Page</title>
<script language="JavaScript">
<!--
      function validateForm ( )
      {
            var op = document.userOp.op.value;
            var type = document.userOp.type.value;
            if(op == "Choose" || type == "Choose")
            {
                  alert("Please select an operation and type to
perform");
                  return false;
            }
            var opType = op + type;
            if((op == "add" || op == "update" || op == "delete") &&
               (type == "Product" || type == "Project"))
               {
                  alert("Only TPTeam Administrators may
add/update/delete Products or Projects.");
                  return false;
               }
               else if((op == "add" || op == "delete") && type ==
"User")
               {
                  alert("Only TPTeam Administrators may add/delete
Users.");
                  return false;
               }
            else if(op == "exec" && (type == "Product" || type ==
"Project" || type == "User"))
               {
                  alert("Only Tests can be executed.");
                  return false;
               }



            document.userOp.action = op + "/" + opType;
            document.userOp.submit();
        return true;
}
```

```
-->
</script>

</head>
<body>

<h3 align="center">TPTeam Home Page</h3>
<hr size="2">
<p>
<form name="userOp" method="post" onSubmit="return validateForm( );">
<P>
<div align="center"><SELECT name="op">
      <OPTION selected value="Choose">Choose Op</OPTION>
      <OPTION value="add">Add</OPTION>
      <OPTION value="update">Update</OPTION>
      <OPTION value="delete">Delete</OPTION>
      <OPTION value="view">View</OPTION>
      <OPTION value="exec">Execute</OPTION>
</SELECT>     <SELECT name="type">
      <OPTION selected value="Choose">Choose Type</OPTION>
      <OPTION value="Product">Product</OPTION>
      <OPTION value="Project">Project</OPTION>
      <OPTION value="Test">Test</OPTION>
      <OPTION value="User">User</OPTION>
</SELECT>
<P><INPUT type="submit" value="Go"></P>
</FORM>
<hr size="2">
</body>
</html>
```

Configuration Files

This section contains the files used to configure TPTeam. First, the Betwixt XML serialization are given. Next, the CSS file used to format the TPManager Web pages is shown. Finally, the TPBridge XMPP connection tpteam.properties file is listed. All Betwixt files are arranged in alphabetical order by name.

chartdatapoint.betwixt

```
<!--
      File        :     chartdatapoint.betwixt

      Author      :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                        for ChartDataPoint objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types as attributes within
      top-level tag
-->
<info primitiveTypes="attribute">
<element name='chartDataPoint'>
<addDefaults/>
</element>
</info>
```

chartdataset.betwixt

```
<!--
      File        :     chartdataset.betwixt

      Author            :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                              for ChartDataSet objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types as attributes within
    top-level tag
-->
<info primitiveTypes="attribute">
<element name='chartDataSet'>
<addDefaults/>
</element>
</info>
```

product.betwixt

```
<!--
      File        :     product.betwixt

      Author      :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                        for Product objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types into the top-level
      tag as attributes
-->
<info primitiveTypes="attribute">
<element name='product'>
<addDefaults/>
</element>
</info>
```

project.betwixt

```
<!--
      File        :     project.betwixt

      Author      :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                        for Project objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types into the top-level
      tag as attributes
-->
<info primitiveTypes="attribute">
<element name='project'>
<addDefaults/>
</element>
</info>
```

test.betwixt

```
<!--
      File        :     test.betwixt

      Author      :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                        for Test objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types into the top-level
      tag as attributes
-->
<info primitiveTypes="attribute">
<!--
      Do not include test executions as part of
      the serialization
-->
<hide property="testExecutions" />
<element name='test'>
<addDefaults/>
</element>
</info>
```

testtype.betwixt

```
<!--
      File        :     testtype.betwixt

      Author      :     Bob Brady, rpbrady@gmail.com

      Contents    :     Betwixt custom serialization file
                        for TestType objects
-->
<?xml version='1.0' encoding='UTF-8' ?>
<!--
      Put all Java primitive types into the top-level
      tag as attributes
-->
<info primitiveTypes="attribute">
<!--
      Do not include all tests having the given
      test type, the test type ID, or test type
      description as part of the serialization
-->
<hide property="tests" />
<hide property="id" />
<hide property="description" />
<element name='testType'>
<addDefaults/>
</element>
</info>
```

add_test_tree.css

```
/*********************************************************************
 *
 * File     :     add_test_tree.css
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     A Cascading Style Sheet that aids in displaying the
 *                TPTeam Test Tree
 *
 *********************************************************************/
.trigger{
      cursor: pointer;
      cursor: hand;
}
.branch{
      display: none;
      margin-left: 16px;
}
```

tpteam.properties

```
#*********************************************************************
#
# File      :     tpteam.properties
#
# Author    :     Bob Brady, rpbrady@gmail.com
#
# Contents  :     A Java Properites file for loading in the TPTeam
#                 XMPPS connection settings and TPEvent topics that
#                 should be handled
#
#
#*********************************************************************
tpmanager.ecfID=tpteam1@gmail.com
tpmanager.password=***
tpbridge.eventadminclient.topics=edu/harvard/fas/rbrady/tpteam/testexec
req,edu/harvard/fas/rbrady/tpteam/projgetreq,edu/harvard/fas/rbrady/tpt
eam/testtreegetreq,edu/harvard/fas/rbrady/tpteam/testdelreq,edu/harvard
/fas/rbrady/tpteam/testdetailreq,edu/harvard/fas/rbrady/tpteam/testupda
tedatreq,edu/harvard/fas/rbrady/tpteam/testupdatereq,edu/harvard/fas/rb
rady/tpteam/testaddreq,edu/harvard/fas/rbrady/tpteam/chartgetdatareq
```

This section contains the database scripts used to create the TPTeam schema, tpteam_ddl.sql, and populate it with seed data, tpteam_dml.sql.

tpteam_ddl.sql

```
/*********************************************************************
 *
 * File     :     tpteam_ddl.sql
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     DDL statements for creating TPTeam schema
 *
 *********************************************************************/

drop database TPTEAM;
create database TPTEAM;
use TPTEAM;
create table TOKEN (USER_ID integer not null, TOKEN_ID text not null,
CREATED_DATE datetime, HOST_IP_ADDR text, primary key (USER_ID,
TOKEN_ID(500)));
create table JUNIT_TEST (ID integer not null, ECLIPSE_HOME text,
WORKSPACE text, PROJECT text, REPORT_DIR text, TPTP_CONNECTION text,
TEST_SUITE text, primary key (ID));
create table PRODUCT (id integer not null auto_increment, NAME text,
DESCRIPTION text, primary key (id), unique index PROD_NAME
(NAME(512)));
create table PROJECT (id integer not null auto_increment, PROD_ID
integer, NAME text, DESCRIPTION text, primary key (id));
create table PROJ_USER (USER_ID integer not null, PROJ_ID integer not
null, primary key (PROJ_ID, USER_ID));
create table ROLE (ROLE_ID integer not null auto_increment, NAME text,
DESCRIPTION text, primary key (ROLE_ID));
create table TEST (id integer not null auto_increment, PROJ_ID integer,
TEST_TYPE_ID integer, CREATED_BY integer, MODIFIED_BY integer,
PARENT_ID integer, NAME text, DESCRIPTION text, IS_FOLDER char(1), PATH
text, CREATED_DATE datetime, MODIFIED_DATE datetime, primary key (id),
unique index TEST_PATH (PATH(512)));
create table TEST_EXECUTION (id integer not null auto_increment, RUN_BY
integer, TEST_ID integer, STATUS char(1), EXEC_DATE datetime, COMMENTS
text, primary key (id));
create table TEST_TYPE (ID integer not null auto_increment, NAME text,
primary key (ID));
create table TPTEAM_USER (id integer not null auto_increment, ROLE_ID
integer, USER_NAME text, FIRST_NAME text, LAST_NAME text, PASSWORD
text, ECF_ID text, PHONE varchar(15), EMAIL text, CREATED_BY integer,
CREATED_DATE datetime, MODIFIED_BY integer, MODIFIED_DATE datetime,
primary key (id));
alter table JUNIT_TEST add index FK28EF29635D877391 (ID), add
constraint FK28EF29635D877391 foreign key (ID) references TEST (id);
```

```
alter table PROJECT add index FK185BD6F93F922B8E (PROD_ID), add
constraint FK185BD6F93F922B8E foreign key (PROD_ID) references PRODUCT
(id);
alter table PROJ_USER add index FK7238936D4A401337 (USER_ID), add
constraint FK7238936D4A401337 foreign key (USER_ID) references
TPTEAM_USER (id);
alter table PROJ_USER add index FK7238936D3F9763F2 (PROJ_ID), add
constraint FK7238936D3F9763F2 foreign key (PROJ_ID) references PROJECT
(id);
alter table TEST add index FK273C923F9763F2 (PROJ_ID), add constraint
FK273C923F9763F2 foreign key (PROJ_ID) references PROJECT (id);
alter table TEST add index FK273C92667B5863 (TEST_TYPE_ID), add
constraint FK273C92667B5863 foreign key (TEST_TYPE_ID) references
TEST_TYPE (ID);
alter table TEST add index FK273C92F6FAB2B5 (MODIFIED_BY), add
constraint FK273C92F6FAB2B5 foreign key (MODIFIED_BY) references
TPTEAM_USER (id);
alter table TEST add index FK273C92D8EE1746 (PARENT_ID), add constraint
FK273C92D8EE1746 foreign key (PARENT_ID) references TEST (id);
alter table TEST add index FK273C92A4A8CE36 (CREATED_BY), add
constraint FK273C92A4A8CE36 foreign key (CREATED_BY) references
TPTEAM_USER (id);
alter table TEST_EXECUTION add index FKCC0C438B1C31CBB3 (RUN_BY), add
constraint FKCC0C438B1C31CBB3 foreign key (RUN_BY) references
TPTEAM_USER (id);
alter table TEST_EXECUTION add index FKCC0C438B8BE725E (TEST_ID), add
constraint FKCC0C438B8BE725E foreign key (TEST_ID) references TEST
(id);
alter table TOKEN add index FK4C4C1D94A401337 (USER_ID), add constraint
FK4C4C1D94A401337 foreign key (USER_ID) references TPTEAM_USER (id);
alter table TPTEAM_USER add index FKCF605571AF9837DE (ROLE_ID), add
constraint FKCF605571AF9837DE foreign key (ROLE_ID) references ROLE
(ROLE_ID);
```

tpteam_dml.sql

```
/*********************************************************************
 *
 * File     :     tpteam_dml.sql
 *
 * Author   :     Bob Brady, rpbrady@gmail.com
 *
 * Contents :     DML statements for inital TPTeam seed data
 *
 *********************************************************************/

USE TPTEAM;

/* Initial Roles, used by web interface */

INSERT INTO ROLE (ROLE_ID, NAME, DESCRIPTION)
      VALUES (1, 'admin', 'Admin role');

INSERT INTO ROLE (ROLE_ID, NAME, DESCRIPTION)
      VALUES (2, 'user', 'Basic User role');

/* Initial administrative user, for web interface
   Note: Passwords are SHA1 hashes */

/* Inital password is "tpteam" for admin user tpteam1 */

INSERT INTO TPTEAM_USER (ID, USER_NAME, PASSWORD, ROLE_ID)
      VALUES (1, 'tpteam1',
'84FE0D304EE0B3D09C860FA4544E17C5D94D29E0',1);

/* Mandatory test types used by TPTeam */

INSERT INTO TEST_TYPE (ID, NAME)
      VALUES (1, 'Folder');

INSERT INTO TEST_TYPE (ID, NAME)
      VALUES (2, 'JUnit');
```