ProtoCom for Java EE Servlets

Installation and Development Guide

Christian Klaussner, updated 28 February 2015

This document explains how to install ProtoCom for Java EE Servlets. Furthermore, it describes the structure of the framework and gives indications for future development.

1 Tips

- Don't use the browser integrated into Eclipse if you are on Windows. It uses Internet Explorer and has problems displaying some parts of the generated prototypes.
- If you want to extend the ProtoCom framework (without changing the transformations), you can speed up the development process by starting an Eclipse instance, creating a test model, and building a performance prototype for it. In your development instance of Eclipse, import the generated prototype. That way you don't need to start a new Eclipse instance each time you change something.

2 Troubleshooting & Known Issues

Problem:
HTTP Status 500 - Guice provision errors with root cause:
java.lang.UnsupportedOperationException: java.lang.reflect.InvocationTargetException
org.palladiosimulator.protocom.framework.java.ee.storage.**EcmProxy**.getSession
...

Solution:
The EcmProxy class connects to the SAP HANA Cloud Document Service. Therefore, you need to start MongoDB with *mongod --dbpath path/to/database* before using the prototype (cf. https://help.hana.ondemand.com/help/frameset.htm?1c6d4a951e7c48c1acfd29b63b56ef43.html)

—

Problem:
Tomcat doesn't start due to a timeout.

Solution:
Increase the timeout by double-clicking on the Java Web Tomcat in the *Servers* view and increasing the *Start* value under the *Timeouts* dropdown.

—

Problem:
Tomcat throws an InstantiationException when loading the main Servlet.

Solution:
The project wasn't deployed correctly. Remove it from the server by choosing „Add and Remove…" from the server's context menu, press „Finish" and deploy the project again.

—

Problem:
In rare cases, sending data from the server to the calibration WebSocket on the client causes the following exception:

```
java.lang.IllegalStateException: The remote endpoint was in state
[TEXT_FULL_WRITING] which is an invalid state for called method
```

Solution:
Unknown…

Browser Issues

- Don't use the browser integrated in Eclipse!
- Safari 8.0.2 shows a warning in the console when loading the prototype:
  `[Warning] Unexpected CSS token: : (font-awesome.min.css, line 4)`
  This is caused by the Internet Explorer compatibility of FontAwesome and can be ignored.
- Safari 8.0.2 shows an error when downloading a JMX or results file:
  `Failed to load resource: Frame load interrupted`
  This seems to be a bug in Safari and can be ignored (all files are downloaded correctly).

3 Future Work

Calibration

The current HDD calibration process writes files to the web server's temporary directory. Alternatively, the SAP Document Service could be used instead to reflect the behavior of actual applications more accurately.

Distributed Systems

The Servlet transformations for ProtoCom were designed to generate prototypes that can be run as distributed system. For example, you would be able to start one instance of a prototype on the SAP HANA Cloud and another instance on a local machine. However, this has not been continuously tested. Furthermore, the transformations currently have no support for variables because this requires `StackContext` objects to be passed around using the RPC protocol. Serialization of these objects is yet to be implemented.

Transformation Refactoring

All dependencies to the JavaNames class (and eventually the class itself) should be removed. This has already been done for most of the Servlet transformations.

Eclipse Dependencies in the Java EE Servlet Framework

The Java EE Servlet framework has dependencies to several Eclipse plug-ins, e.g., to the StoEx parser and EMF. In order to use these plug-ins in generated performance prototypes, their respective JAR files have to be copied to the generated Java EE project each time a PCM instance is transformed. This copying of files is managed by the `FileProvider` class in the `webcontent` package. The `index` file in the same package lists all files that are copied (including JavaScript, CSS, etc.; plug-ins are listed at the end). The problem with this approach is that changes in the plug-ins are not automatically included in the framework because their updated JAR files have to be copied manually to the framework. In the future, it would be better to retrieve the JAR files of the plug-ins from the currently running Eclipse instance.

Overlap between Java SE and Java EE Framework

Currently, the Java SE and Java EE frameworks are completely separated. However, there is potential for refactoring the Java SE transformations and merging the Java SE framework with Java EE. For example, both frameworks use the same machanism for taking time measurements.

Replacing the Sensor Framework

The Java EE Servlet framework is designed to support different measurement frameworks through dependency injection and the `IExperiment` interface. That way, the Sensor Framework can be easily replaced by other technologies. Regarding Java SE, however, the Sensor Framework is deeply integrated into the framework and has to be separated from it.

4 Implementation

PCM Adapters

The Java EE Servlet transformations refrain from using the PCM EMF model directly to retrieve information. Instead, they use the adapter classes contained in the `model` package which provide a more readable interface to the required data. However, these adapters are currently used only by the Servlet transformations. Other transformations should be updated to also use the adapters.

Note that the functionality provided by the adapters is not complete yet. If additional data needs to be retrieved from the model, please update the appropriate adapters instead of using the model directly.

Package Structure

See the table at the end of this document.

Dependency Injection

The framework manages dependencies using dependency injection with Google Guice. The module configuration can be found in `main/ProtoComModule.java`. When extending the framework with new experiment and storage mechanisms as described in section *Package Structure*, this module has to be changed to inject the new dependencies. The following two lines control the implementations used for `IStorage` and `IExperiment`:

```
bind(IStorage.class).to(EcmStorage.class);
bind(IExperiment.class).to(SensorFrameworkExperiment.class);
```

`api`	Contains Jersey and WebSocket classes for the HTTP API. The front end of the generated performance prototypes uses the functionality provided by these classes to interact with the framework. The WebSockets are used for pushing real-time data from the framework to the prototypes, e.g., log messages and calibration status updates.
`experiment`	Contains classes for conducting experiments. Currently, the only implemented experiment mechanism uses the Sensor Framework. However, other means of measurement can be integrated by implementing the `IExperiment` interface and configuring the framework accordingly (see *Dependency Injection*).
`main`	Contains the prototype entry point, dependency injection module, logging mechanism and a JSON helper class.
`modules`	Contains module classes that are used for managing startable modules, i.e., systems an resource containers generated from the PCM. Information about these modules is presented in the front end when running performance prototypes.
`protocol`	Contains the implementation of the HTTP RPC protocol and the object registry.
`prototype`	Contains classes and interfaces used directly by generated prototypes (e.g., base classes like `PortServlet`). The `PrototypeBridge` class provides a central point for the prototype to transfer model-specific information, e.g., processing rates, to the framework. A single instance of this class is created by the framework during startup and then passed to the prototype which fills it with the required data.
`storage`	Contains classes for storing files in Java EE environments. Currently, the only implemented mechanism uses the SAP HANA Cloud Document Service. However, other means of storage can be integrated by implementing the `IStorage` interface an configuring the framework accordingly (see *Dependency Injection*).
`webcontent`	Contains file required by generated prototypes.