

ObjectWeb - Annual Conference 2006 - UseCase - No7

[No7](#)

UC7: Embedded Workflow Engine - usage of Shark and JaWE by AKBANK

Use Case	
ObjectWeb components	Shark, JaWE
Other components	
Abstract	<p>Intense and mission critical production usage of Shark as embedded server side workflow engine. Approval mechanism of AKBANK's retail banking project rely on Shark, taking the load of approval mechanism from the host. Shark working in a 8 clone environment without any failover problems and zero coupling of Shark observed with server side application. 5,000 processes processed per day and 20,000 processes planned.</p> <p>Preliminary approval mechanism adapted and replaced with Shark Engine easily. Extended classes : DODSUserGroupManager, HistoryRelatedAssignmentManager, ObjectFactory, PackageAdmin, UserGroupAdmin, UserGroupQuery, WfActivityImpl and related interfaces. 'Assignment explosion' problem solved by adding branch code property to the users and extending the assignment mechanism.</p> <p>Production problem solved quickly and effectively by optimizing Shark.conf and Shark Teams' support resulted in a %200 performance gain.</p> <p>Pre-checking of approval falling conditions in a separate mechanism is a necessity for high volume online transacting systems. Shark is capable of handling large number of processes , but precondition checking inside Shark is unnecessary especially under a heavy online transaction load.</p>
Details	<p>At the beginning of 2004, we were seeking a way to improve our preliminary implementation of approval mechanism in Akbank's retail branch project. We asked our technical people for prior workflow experience and products. There were some paid products at hand that have never been implemented.</p> <p>We decided to use an open source workflow engine and editor, rather than trying to write our own or buying another product which we can never be sure of its suitability beforehand. As we continued our search on the internet, we met JaWE and Shark. At the first sight, we loved JaWE and begun to enrich our workflow experience. We were instantly 'playing' with our flows and happy to see that they were working the way we think of.</p> <p>After a few weeks of investigation, we were convinced that JaWE and Shark together are mature products that can fulfill our requirements.</p> <p>We downloaded the source code and understood the internals of Shark. We were novice open-source users and surprised to see that open-source projects are built upon other ones!</p> <p>We noticed that, we were under the threat of 'assignment explosion' due to existence of a large number of branches</p>

	<p>and the same participant-role mappings for each branch. By default, an in-branch approval will create assignments for all other branches' users. So we added branch code property for the users and extended the assignment mechanism to use it.</p> <p>We also decided to use Shark UserGroup mechanism and tables without unifying them with our own, something possible with open-source software, especially Sharks' underlying DODS layer. This was a performance decision due to the heavy load expectation on the unified tables. As a result, we implemented a master-slave relationship between the UserGroup mechanisms and synchronized the changes. It was interesting to us that, participants of a process together with the participant-role(group) mappings were conceptually the tasks to authorize for a role in our branch application.</p> <p>We integrated Shark to our server side branch project as an embedded workflow engine. During integration, we used facade pattern and put another layer between Shark and our server project. Thus, we've easily implemented required functionalities, taking care of changes in the interface. We do not use locking mechanism and audits for performance issues. We adapted our preliminary approval mechanism and interfaced it with Shark Engine easily. Extended classes are, DODSUserGroupManager, HistoryRelatedAssignmentManager, ObjectFactory, PackageAdmin, UserGroupAdmin, UserGroupQuery, WfActivityImpl and related interfaces.</p> <p>Before we go into production, we did extensive stress testing and measured Shark interaction times. We were satisfied with the testing and decided to go into production. The big day came and on a monday morning, we were in production. We saw that, after 5 minutes, the system 'exploded'. That day we by-passed approval mechanism, and examined the performance problem. Needless to say that, by-passing the approval mechanism meant that the bank was facing the risk of high amountted transactions without control, which was unacceptable. We quickly optimized the sytem using Shark.conf and pre-checked the approval falling conditions in a seperate javascript mechanism, assuring that only the 'true' processes creating instances in Shark. Numerically, this meant that only 50 new processes are arriving to Shark as compared to 1000 new instances per minute. We learned that, one should not put his workflow engine in front of a heavy online transaction load. Instead a precondition checking mechanism will prevent unnecessary loading. We also e-mailed our problem to the Shark Team. After 15 minutes, we received a reply call. The next day and forth is a success story. We also got Shark Teams' advices and applied them, resulting in a %200 performance gain.</p> <p>Our environment consists of :</p> <ul style="list-style-type: none">• 4 Unix machines with AIX 5.2 and WAS 5.1 installed on each. <p>2 clones on each machine.</p> <ul style="list-style-type: none">• Seperate database machine (IBM UDB 7.2)and seperate database instance for Shark persistence. <p>Application characteristics:</p> <ul style="list-style-type: none">• 600 branches, 6000 users.• 200,000 monetary operations (7 different functions, using 3 different flows) subject to approval conditions
--	--

ObjectWeb - Annual Conference 2006 - UseCase - No7

	<p>that result in 5,000 processes(satisfying approval falling conditions) and 12,000 activities(approval steps) per day.</p> <ul style="list-style-type: none">• With ongoing development of new approval functions, volume expected to reach 20,000 processes per day.• 200 ms average response time for each Shark interaction.• Nearly zero coupling with application and WAS, meaning that Shark is working whether or not application is! <p>Zero Shark problems, zero Shark downtime in a 4 month prod period.</p>
Additional materials	<ul style="list-style-type: none">• D:desktopWORKFLOWCON2006Screenshots.doc
Why is this use case a success?	
<p>1) Mission critical production use under heavy load. 2) Performance problem solved easily by optimization. 3) Further performance gain obtained by utilizing advices of Shark team.(Very short response time) 4) Existing preliminary approval mechanism efficiently replaced with Shark, bringing much more flexibility. 5) Using a priced product would cost 100 times higher! (We have 20 CPU's in production) 6) Shark code customized as needed.</p>	

Public Contact	
Organization	AKBANK
Country	Turkey

[No7](#) (en)

Creator: Date: 2005/12/15 13:19

Last Author: xwiki:XWiki.moghrabi Date: 2006/01/11 15:35

Copyright (c) 2005-2006, [ObjectWeb Consortium](#)