XOOPS 2.6.0 Service Manager: Introduction - XOOPS

NEWS_PDF_AUTHOR: Mamba

NEWS_PDF_DATE: 2014/5/21 7:53:57

In XOOPS 2.6.0 alpha 2, some familiar services that were traditionally internal parts of the core were separated into modules. Some examples are: avatars, comments, and notifications The separated module approach achieves some important benefits: ? Modules can be updated independently. ? Modules can have private resources, such as templates, configurations, maintenance pages. ? Modules can be omitted if not needed, saving some resources. But there are potential benefits to separation that were not realized: ? The service modules were not easily replaced with alternate implementations ? References using hard coded module names litter the entire system wherever a service is needed In XOOPS 2.6.0 we'll introduce a "**Service Manager**" component: ? Services located by service name, not provider ? Service interface established by Contract ? Returns a standardized Response object that includes result, status and messages ? Request is based on a well known interface ? Actual provider does not matter to caller ? No need to check for a specific module ? If the service is not available, that status is returned just like any other error condition. ? Service providers are only instantiated when explicitly requested, and then kept for the duration of the PHP run. ? A locate event is not triggered until a named service is first requested, so if a service is not used, it has no overhead cost. ? If no providers for a service are installed, the locate trigger has little cost, and any subsequent calls go straight to a NullProvider that minimizes resource use. **Richard Griffith**, our Core Team leader, has created a presentation to show you how XOOPS 2.6.0 will implement the Service Manager, and how to use it. You can see the presentation on **SlideShare** This is the second presentation from our new "**XOOPS 2.6.0 Education Series**", where we will share with new features being developed for XOOPS 2.6.0, and show you how to use them. The first presentation was about Asset Management using Assetic **XOOPS 2.6.0** is currently in development, in pre-Alpha 3 stage. There is a lot of new development going into XOOPS 2.6.0, as you can see from these articles: XOOPS is back, and with a vengeance!

Major improvements to XOOPS 2.6.0, Alpha 3 release is getting closer! and after adding **Doctrine**, **PHPUnit**, **Composer**, **Assetic**, and several other cool things, **Service Manager** is another component that will simplify XOOPS development! **You can contribute to the XOOPS 2.6.0 development by forking it from GitHub, and submitting your code there.** *We would also like to thank **Slider84** from XOOPS France for his help in creating the template for this presentation!* Viva XOOPS!

In XOOPS 2.6.0 alpha 2, some familiar services that were traditionally internal parts of the core were separated into modules. Some examples are: avatars, comments, and notifications The separated module approach achieves some important benefits: ? Modules can be updated independently. ? Modules can have private resources, such as templates, configurations, maintenance pages. ? Modules can be omitted if not needed, saving some resources. But there are potential benefits to separation that were not realized: ? The service modules were not easily replaced with alternate implementations ? References using hard coded module names litter the entire system wherever a service is needed In XOOPS 2.6.0 we'll introduce a "**Service Manager**" component: ? Services located by service name, not provider ? Service interface established by Contract ? Returns a standardized Response object that includes result, status and messages ? Request is based on a well known interface ? Actual provider does not matter to caller ? No need to check for a specific module ? If the service is not available, that status is returned just like any other error condition. ? Service providers are only instantiated when explicitly requested, and then kept for the duration of the PHP run. ? A locate event is not triggered until a named service is first requested, so if a service is not used, it has no overhead cost. ? If no providers for a service are installed, the locate trigger has little cost, and any subsequent calls go straight to a NullProvider that minimizes resource use. **Richard Griffith**, our Core Team leader, has created a presentation to show you how XOOPS 2.6.0 will implement the Service Manager, and how to use it. You can see the presentation on **SlideShare** This is the second presentation from our new "**XOOPS 2.6.0 Education Series**", where we will share with new features being developed for XOOPS 2.6.0, and show you how to use them. The first presentation was about Asset Management using Assetic **XOOPS 2.6.0** is currently in development, in pre-Alpha 3 stage. There is a lot of new development going into XOOPS 2.6.0, as you can see from these articles: XOOPS is back, and with a vengeance! 

Major improvements to XOOPS 2.6.0, Alpha 3 release is getting closer! and after adding **Doctrine**, **PHPUnit**, **Composer**, **Assetic**, and several other cool things, **Service Manager** is another component that will simplify XOOPS development! **You can contribute to the XOOPS 2.6.0 development by forking it from GitHub, and submitting your code there.** *We would also like to thank **Slider84** from XOOPS France for his help in creating the template for this presentation!* Viva XOOPS!