POSTSHARP

CASE STUDY:
Gamesys

"If we were able to continuously develop and operate several high-volume games in production with such a small team, it was partly thanks to PostSharp."

Yan Cui
Senior Backend Developer
Gamesys

Gamesys

The market leader in the field of real money gaming improves productivity and reduces development and maintenance costs with PostSharp.

> *"Thanks to PostSharp, we were able to localize over 95 percent of the game with just one line of code and save potentially hundreds of man hours over the lifetime of the game."*
>
> **Yan Cui**
> *Senior Backend Developer*
> *Gamesys*

Overview

Gamesys serves around one million daily active users across its social games. Their backend services handle more than 250 million requests per day. Despite its massive scale, this distinctive service is being maintained by a remarkably small development team of just seven super-productive individuals. They chose PostSharp to automate the implementation of their design patterns, outsourcing mundane tasks to the compiler so they can focus on what matters.

Small team, big challenges

Although Gamesys employs more than 1,000 people globally and operates in a number of markets across the UK, Europe and the US, the Social team's backend services are overseen by a small team of developers. As such, it is critical that the team minimize inconveniences and concentrate on tasks that are essential to the work at hand.

"With a small team of seven developers who are responsible for everything that happens on the back end, it's imperative that we allow them to focus on things that add value to our players and maximise their productivity," says Yan Cui, Senior Backend Developer at Gamesys's Social team.

The team is responsible for building scalable backend services to support Gamesys' social games on mobile devices and Facebook. Across the company's range of social games, it has around one million daily active users, and its backend services receive around 250 million requests per day. This requires the team to deliver high-quality solutions that will enable the backend team to work at peak efficiency.

The team has worked on several projects that presented various challenges and called for comprehensive problem-solving approaches:

1. Performance monitoring;
2. Error handling; and
3. Localization.

Why Gamesys chose PostSharp

The team chose PostSharp over alternative solutions for a number of reasons:

- PostSharp offers the most complete support of features they have been looking for, such as:
  - The ability to multicast aspects using filters, therefore targeting several methods employing a single line of code;
  - The ability to intercept events; and
  - The ability to introduce members dynamically.
- PostSharp provides the best runtime performance, as most of the work is done at compile time.
- PostSharp works consistently across public, private and static members and does not depend on an object that is constructed by the IOC container.
- PostSharp enables developers to automatically verify at build time how aspects are being used, failing the build if an aspect has been applied to an inadequate method.

"When I first joined Gamesys in 2010, our code base was littered with cross-cutting concerns such as exception handling and validation, and in general the control flows were so convoluted that it was difficult to see what the application was actually doing a lot of the time," says Yan. "So when I encapsulated those repeated patterns into several aspects and cleaned up our code base using PostSharp, it was all the motivation and convincing everyone needed."

Performance Monitoring: Solved

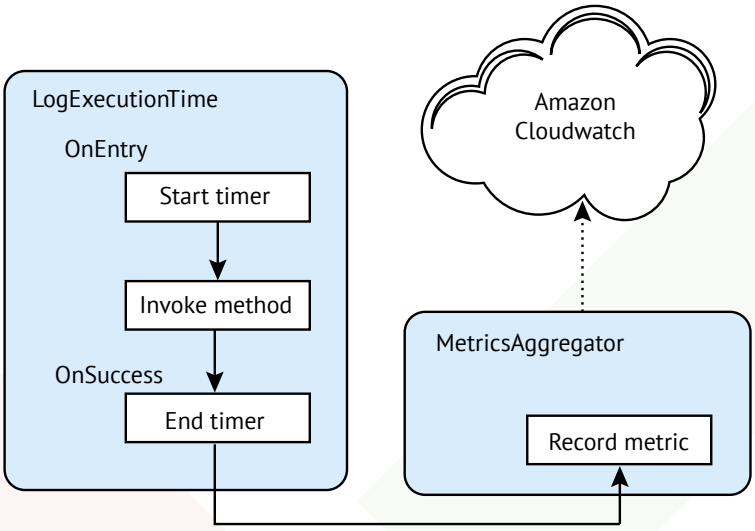
For Gamesys, it is essential to have visibility of both what and how their application is doing. Therefore, the company needed a solution that would make it easy for its developers to track execution time and count of service entry points, IO operations, and CPU-intensive methods.

Taking a traditional approach to solve this would have led to a lot of boilerplate code and would have cluttered the company's code base.

With PostSharp, the team created a pair of aspects – *LogExecutionTime* and *LogExecutionCount* – to track the execution time and the count of any performance-sensitive method.

> *"Over the years, Postsharp has helped us to save over tens of thousands lines of code ."*
>
> **Yan Cui**
> *Senior Backend Developer*
> *Gamesys*

The pair of aspects records execution metrics pertaining to these methods and funnels them to an in-memory aggregator that aggregates the per-instance metrics and publishes them to *Amazon CloudWatch* periodically.



Error Handling: Solved

Effective error handling presented a real challenge for Gamesys, as the company had hundreds of specific errors for each of its games.

Every request to the company's client-facing services has a matching response object, which derives from a *BaseResponse* class that looks like the following:

```
public abstract BaseResponse
{
  public int? ErrorCode { get; set; }
  public string ErrorMessage { get; set; }
}
```

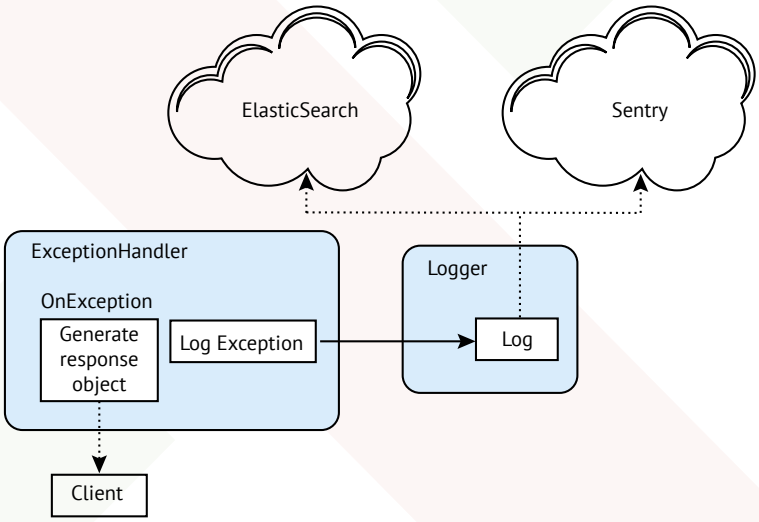
When a request fails for whatever reason (e.g., a player tries to buy a Broadaxe but has run out of space in his or her backpack), they always return a response object back to the client with an *ErrorCode* so that the client can address it depending on the type of error.

The team created a custom PostSharp aspect, an *ExceptionHandler*, which provides team members with desired functionality, and which is now a core piece in the company's infrastructure.

The team multicast the aspect to all methods that are service entry points so that anytime an exception is thrown and bubbles up, the aspect will:

1. Capture the exception information;
2. Log it with Gamesys' logging infrastructure;
    a. Record all exceptions in the *ElasticSearch* cluster;
    b. Push critical (all the custom exceptions specify a *SeverityLevel*) exceptions that warrant more urgent attention to Sentry, which sends notifications to the relevant team members;
3. Create a response object of the correct type;
4. Populate the *ErrorCode* and *ErrorMessage* properties; and
5. Return the dynamically created response object.

"This simple aspect is reused in every project and plays an important role in our infrastructure," says Yan.



Localization: Solved

"It was something of a mammoth task when we decided to localize the game *Here Be Monsters* for Flash and iOS clients, as we have more than 3,500 items and 1,500 quests in the game, with more text than the first three Harry Potter books combined," says Yan.

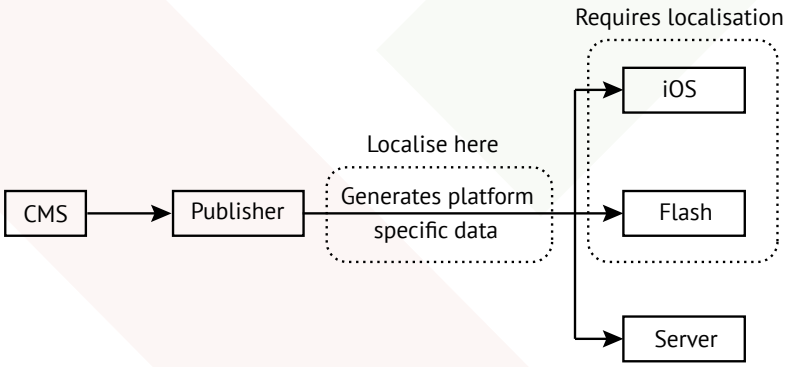
With the conventional approach, the client would consume a *gettext* file containing all of the translations, and anywhere some text needs to display, the *gettext* file would substitute the original text with the localized text.

This would lead to several issues:

- A large number of code files need to change during implementation.
- Maintenance overhead - all future changes need to take localization into account.
- Duplicated efforts - need to replicate changes across client platforms.
- Hard to scale - complicates implementation and testing; easy for regression to creep in during frequent release cycles.

To localize the game *Here Be Monsters* for Flash and iOS clients, the team created custom aspects with PostSharp and performed localization on the server as part of the pipeline that validates and publishes the data (quests, achievements, items, etc.) captured in their custom CMS.

> *"If we were able to continuously develop and operate several high-volume games in production with such a small team, it was partly thanks to PostSharp."*
>
> **Yan Cui**
> *Senior Backend Developer*
> *Gamesys*



As part of its solution, the *Publisher* would:

1. Ingest the *gettext* translation file.
2. Use the *Localize* aspect (see a simplified version here) to intercept *string* property setters on DTOs to replace the input string with the localized version.
3. Repeat for each language to generate a language-specific version of the DTOs.

To automatically apply localization to all present and future DTO types, they simply multicast the attribute and target all types that follow our naming convention for DTOs:

```
[assembly: Localize(AttributeTargetTypes = "*DTO")]
```

"With one line of code, we were able to localize over 95 percent of the game and save potentially hundreds of man hours over the lifetime of the game," says Yan.

Summary

The team at Gamesys is pleased with the results from using PostSharp:

- Improved productivity thanks to more concise and maintainable code;
- Less code to write, read and maintain, resulting in reduced development and maintenance costs; and
- Fewer bugs.

"It's difficult to measure how many lines of code and how much development and maintenance costs PostSharp has helped us to save, but over the years, we could have easily saved over tens of thousands lines of code," says Cui. "The fact that we are able to continuously develop and operate several high-volume games in production with such a small team is in part thanks to PostSharp."

SharpCrafters s.r.o.
Namesti 14 rijna, 1307/2
150 00 Prague 5
Czech Republic

US: +1 866 576 5361
CZ: +420 270 007 790
www.postsharp.net
info@postsharp.net