🌳 Spring AI PR Orchard Dashboard

Run: run-22 PRs Analyzed: 16 Generated: 2025-08-21 01:35:59

📦 **7** Backport Ready

🍎 **5** Ready to Harvest

✅ **12** Low Risk PRs

📊 **16** Total PRs

🍎 Ready to Harvest

Low Hanging Fruit - Quick Wins (5 PRs)

🍎 #4156: Fix reordering issue in AzureOpenAiChatModel (Fixes #4155)

by **robinmayerhofer** Type: Bugfix Risk: LOW

Backport: ✅ APPROVE

Files:	Lines:	Effort:
1	+2/-2	1/10

💡 Problem Summary

The AzureOpenAiChatModel has a race condition in streaming responses where chat chunks can arrive out of order due to Flux::flatMap not providing ordering guarantees. This causes streamed content like 'data: <A>' followed by 'data: <B>' to potentially be received as '<B> => <A>'.

- Maintain strict ordering of streaming chat response chunks
- Preserve existing Spring AI chat model API contract and behavior
- Ensure compatibility with Azure OpenAI streaming protocol
- *+ 5 more requirements...*

View on GitHub 🔀 Checkout & Merge 🔍 View Assessments

🍎 #4161: doc:Update README.md

by **YunKuiLu** Type: Documentation Risk: LOW

Backport: ✅ APPROVE

Files:	Lines:	Effort:
1	+5/-0	1/10

💡 Problem Summary

The Spring AI project lacks visibility for community resources, making it difficult for users to discover the Spring AI Community organization and related community-driven resources. This PR aims to improve community discoverability by adding a direct link to the Spring AI Community GitHub organization.

- Add clear link to Spring AI Community organization in README
- Maintain consistency with existing community resource section formatting
- Ensure link placement follows logical flow with other community resources
- *+ 2 more requirements...*

View on GitHub 🔀 Checkout & Merge 🔍 View Assessments

🍎 #3999: fix: GH-3998 Refactor the `ollamaChatRequest` method of `OllamaChatModel`

by **sunyuhan1998** Type: Bugfix Risk: LOW

Backport: ✅ APPROVE

Files:	Lines:	Effort:
1	+14/-13	2/10

💡 Problem Summary

The OllamaChatModel uses direct instanceof checks to identify message types, which prevents users from implementing custom AbstractMessage subclasses and creates inconsistency with other ChatModel implementations like OpenAiChatModel and DeepSeekChatModel that use MessageType-based identification.

- Enable support for custom AbstractMessage implementations
- Maintain consistency with other ChatModel implementations in the Spring AI framework
- Use MessageType enum for message type identification instead of instanceof checks
- *+ 5 more requirements...*

🍎 #4020: Cleanup: Remove unused CallPromptResponseSpec and StreamPromptResponseSpec

by **YunKuiLu** Type: Cleanup Risk: LOW

Backport: ✅ APPROVE

Files:	Lines:	Effort:
1	+0/-18	2/10

💡 Problem Summary

This PR addresses code cleanup by removing two unused interfaces (CallPromptResponseSpec and StreamPromptResponseSpec) from the ChatClient class that are no longer referenced or needed in the codebase, reducing technical debt and maintaining clean API surface.

- Remove unused CallPromptResponseSpec interface without breaking existing functionality
- Remove unused StreamPromptResponseSpec interface safely
- Maintain backward compatibility for existing ChatClient API usage
- *+ 4 more requirements...*

View on GitHub 🔀 Checkout & Merge 🔍 View Assessments

View on GitHub

🔀 Checkout & Merge

🔍 View Assessments

🍎

#4112: GH-4089 Fixed the issue where sens words in chat memory repeatedly triggered SafeGuardAdvisor interception

by **sunyuhan1998** Type: Bugfix Risk: **LOW**

Backport: ✅ **APPROVE**

Files:	Lines:	Effort:
2	**+12/-2**	**2/10**

💡 **Problem Summary**

The SafeGuardAdvisor has a lower default order than MessageChatMemoryAdvisor, causing sensitive content stored in chat memory to repeatedly trigger SafeGuardAdvisor interception on subsequent messages, even when new messages contain no sensitive content. This creates a persistent blocking behavior that degrades user experience.

- SafeGuardAdvisor must execute before MessageChatMemoryAdvisor to prevent sensitive content from being stored in memory
- Default ordering must work correctly without requiring manual configuration by users
- Existing functionality of both advisors must remain unchanged
- *+ 4 more requirements...*

View on GitHub

🔀 Checkout & Merge

🔍 View Assessments

🍃 Easy Picks

Simple Reach - Low-Medium Effort (4 PRs)

🍃

#4148: feat(google genai): support sending labels with chat request

by **garethjevans** Type: Feature Risk: **LOW**

Backport: ❓ **UNKNOWN**

Files:	Lines:	Effort:
3	**+67/-4**	**3/10**

💡 **Problem Summary**

This PR adds support for sending custom labels with Google Gemini AI chat requests in Spring AI. The labels feature allows developers to attach metadata key-value pairs to their AI requests for organizational tracking, billing categorization, or request classification purposes.

- Add labels field as Map<String, String> to GoogleGenAiChatOptions class
- Implement proper merging of runtime and default label options using ModelOptionsUtils
- Ensure labels are properly propagated through the request building process
- *+ 5 more requirements...*

View on GitHub

🔀 Checkout & Merge

🔍 View Assessments

🍃

#4065: refactor(memory): Improve Chat Memory logic and efficiency

by **juntae6942** Type: Feature Risk: **LOW**

Backport: ❌ **REJECT**

Files:	Lines:	Effort:
4	**+127/-26**	**4/10**

💡 **Problem Summary**

The MessageWindowChatMemory implementation had performance overhead because it replaced the entire message list on every update, which is inefficient for long-running conversations. The ChatMemoryRepository API only supported full overwrite operations rather than incremental updates.

- Introduce incremental update capability to avoid full message list replacement
- Abstract storage layer through ChatMemoryRepository interface for future persistent implementations
- Maintain backward compatibility with existing ChatMemoryRepository implementations
- *+ 5 more requirements...*

View on GitHub

🔀 Checkout & Merge

🔍 View Assessments

#4110: [Enhancement] - Enable passing of requestMetadata to the Bedrock Converse

by **StuartLox** Type: Feature Risk: **LOW**

Backport: ❌ **REJECT**

Files:	Lines:	Effort:
5	**+49/-6**	**4/10**

💡 Problem Summary

Spring AI currently doesn't expose a way to pass custom requestMetadata to the AWS Bedrock Converse API, which prevents developers from including important audit and tracing information like correlation IDs and user session IDs in CloudWatch logs for compliance and operational purposes.

- Enable passing custom requestMetadata to Bedrock Converse API calls
- Leverage existing metadata field from user prompt objects as the data source
- Maintain backward compatibility with existing Spring AI Bedrock integration
- *+ 5 more requirements...*

View on GitHub

🔀 Checkout & Merge

🔍 View Assessments