

JSP & Servlets
19 уровень, 5 лекция

ОТКРЫТА

Вот мы и подобрались к самой большой части пакета. Здесь будут описаны интерфейсы для запуска асинхронных задач с возможностью получения результатов через Future и Callable интерфейсы, а также сервисы и фабрики для создания thread pools: ThreadPoolExecutor, ScheduledThreadPoolExecutor, ForkJoinPool.

`Future<V>` — замечательный интерфейс для получения результатов работы асинхронной операции. Ключевым методом здесь является метод `get`, который блокирует текущий поток (с таймаутом или без) до завершения работы асинхронной операции в другом потоке. Также дополнительно существуют методы для отмены операции и проверки текущего статуса. В качестве имплементации часто используется класс `FutureTask`.

`Future<V>` — замечательный интерфейс для получения результатов работы асинхронной операции. Ключевым методом здесь является метод `get`, который блокирует текущий поток (с таймаутом или без) до завершения работы асинхронной операции в другом потоке. Также дополнительно существуют методы для отмены операции и проверки текущего статуса. В качестве имплементации часто используется класс `FutureTask`.

`RunnableFuture<V>` — если `Future` – это интерфейс для Client API, то интерфейс `RunnableFuture` уже используется для запуска асинхронной части. Успешное завершение метода `run()` завершает асинхронную операцию и позволяет вытаскивать результаты через метод `get`.

`Callable<V>` — расширенный аналог интерфейса `Runnable` для асинхронных операций. Позволяет возвращать типизированное значение и кидать `checked exception`. Несмотря на то, что в этом интерфейсе отсутствует метод `run()`, многие классы `java.util.concurrent` поддерживают его наряду с `Runnable`.

`FutureTask<V>` — имплементация интерфейса `Future/RunnableFuture`. Асинхронная операция принимается на вход одного из конструкторов в виде `Runnable` или `Callable` объектов. Сам же класс `FutureTask` предназначен для запуска в `worker` потоке, например, через `new Thread(task).start()` или через `ThreadPoolExecutor`. Результаты работы асинхронной операции вытаскиваются через метод `get(...)`.

`Delayed` — используется для асинхронных задач, которые должны начаться в будущем, а также в `DelayQueue`. Позволяет задавать время до начала асинхронной операции.

`ScheduledFuture<V>` — маркерный интерфейс, объединяющий `Future` и `Delayed` интерфейсы.

`RunnableScheduledFuture<V>` — интерфейс, объединяющий `RunnableFuture` и `ScheduledFuture`. Дополнительно можно указывать является ли задача одноразовой или же должна запускаться с заданной периодичностью.

[← Предыдущая лекция](#)

[Следующая лекция →](#)

 +6 

Комментарии

популярные новые старые

JavaCoder

Введите текст комментария



У ЭТОЙ СТРАНИЦЫ ЕЩЕ НЕТ НИ ОДНОГО КОММЕНТАРИЯ

ОБУЧЕНИЕ

- Курсы программирования
- Курс Java
- Помощь по задачам
- Подписки
- Задачи-игры

СООБЩЕСТВО

- Пользователи
- Статьи
- Форум
- Чат
- Истории успеха
- Активности

КОМПАНИЯ

- О нас
- Контакты
- Отзывы
- FAQ
- Поддержка



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

