**private** **static** **final** Log ***log*** = LogFactory.*getLog*(TestAsyncController.**class**);

/\*\*

\* callable处理其实是在WebAsyncManager内部封装成WebAsyncTask后再处理

\*

\* **@param** msg

\* **@return**

\*/

@RequestMapping("callable")

**public** Callable<ResultModel> callable(String msg) {

**return** () -> {

***log***.info("当前线程为：" + Thread.*currentThread*().getName());

Thread.*sleep*(1000);

**return** **new** ResultModel(**true**, "callable成功", msg);

};

}

@RequestMapping("webAsyncTask")

**public** WebAsyncTask<ResultModel> webAsyncTask(String msg) {

**long** timeout = 10000L;

WebAsyncTask<ResultModel> task = **new** WebAsyncTask<>(timeout, () -> {

***log***.info("当前线程为：" + Thread.*currentThread*().getName());

Thread.*sleep*(5000);

**return** **new** ResultModel(**true**, "webAsyncTask成功", msg);

});

**return** task;

}

/\*\*

\* DeferredResult是用于保存延迟处理结果的类，当一个处理器返回DeferredResult类型的返回值时，将启动异步处理，表示将结果交给别的线程

\*

\* **@param** msg

\* **@return**

\*/

@RequestMapping("deferred")

**public** DeferredResult<ResultModel> deferred(String msg) {

DeferredResult<ResultModel> result = **new** DeferredResult<>(10000L);

approve(result, msg);

**return** result;

}

**private** **void** approve(DeferredResult<ResultModel> result, String msg) {

**new** Thread(() -> {

**try** {

Thread.*sleep*(5000);

***log***.info("当前线程为：" + Thread.*currentThread*().getName());

result.setResult(**new** ResultModel(**true**, "测试DeferredResult", msg));

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}).start();

}