# Netflix Zuul快速入门

## Zuul的Maven依赖

<dependency>

<groupId>com.netflix.zuul</groupId>

<artifactId>zuul-core</artifactId>

<version>1.0.0</version>

</dependency>

简单示例项目[zuul-simple-webapp](https://github.com/Netflix/zuul/wiki/zuul-simple-webapp)

zuul-simple-webapp是一个示例的web应用程序，它显示了一些zuul-core的简单用例。

理解它的最好方法是将其加载到您的环境中，运行它，修改它，然后再次运行以注意更改。

## web.xml 声明

所在目录：zuul-simple-webapp\src\main\webapp\WEB-INF\web.xml

Web.xml的所有内容：

<listener>

<listener-class>com.netflix.zuul.StartServer</listener-class>

</listener>

<servlet>

<servlet-name>Zuul</servlet-name>

<servlet-class>com.netflix.zuul.http.ZuulServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Zuul</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

<filter>

<filter-name>ContextLifecycleFilter</filter-name>

<filter-class>com.netflix.zuul.context.ContextLifecycleFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>ContextLifecycleFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

StartServer作为初始化应用程序的ServletContextListener。

ZuulServlet是一个匹配所有请求的servlet。 它执行执行前置，路由和后置过滤器的核心Zuul过滤器流程。

ContextLifecycleFilter是一个匹配所有请求的servlet过滤器。 它在每个请求后清理RequestContext，确保隔离。

## StartServer 初始化

StartServer首先将FilterLoader的编译器设置为GroovyCompiler实例。

FilterLoader.getInstance().setCompiler(new GroovyCompiler());

可以将编译器设置为能够将文件或命名的代码段转换为类实例的DynamicCodeCompiler的任何实现。

然后，我们将初始化FilterFileManager，以便每5秒钟在指定的前，路由和发布目录中的文件系统上查找Groovy文件。

final String scriptRoot = System.getProperty("zuul.filter.root");

try {

FilterFileManager.setFilenameFilter(new GroovyFileFilter());

FilterFileManager.init(5,

scriptRoot + "/pre",

scriptRoot + "/route",

scriptRoot + "/post");

} catch (Exception e) {

throw new RuntimeException(e);

}

## 过滤器执行流程