# フレームワーク基礎実習

# 10. Webフィルター

株式会社ジードライブ

# 今回学ぶこと

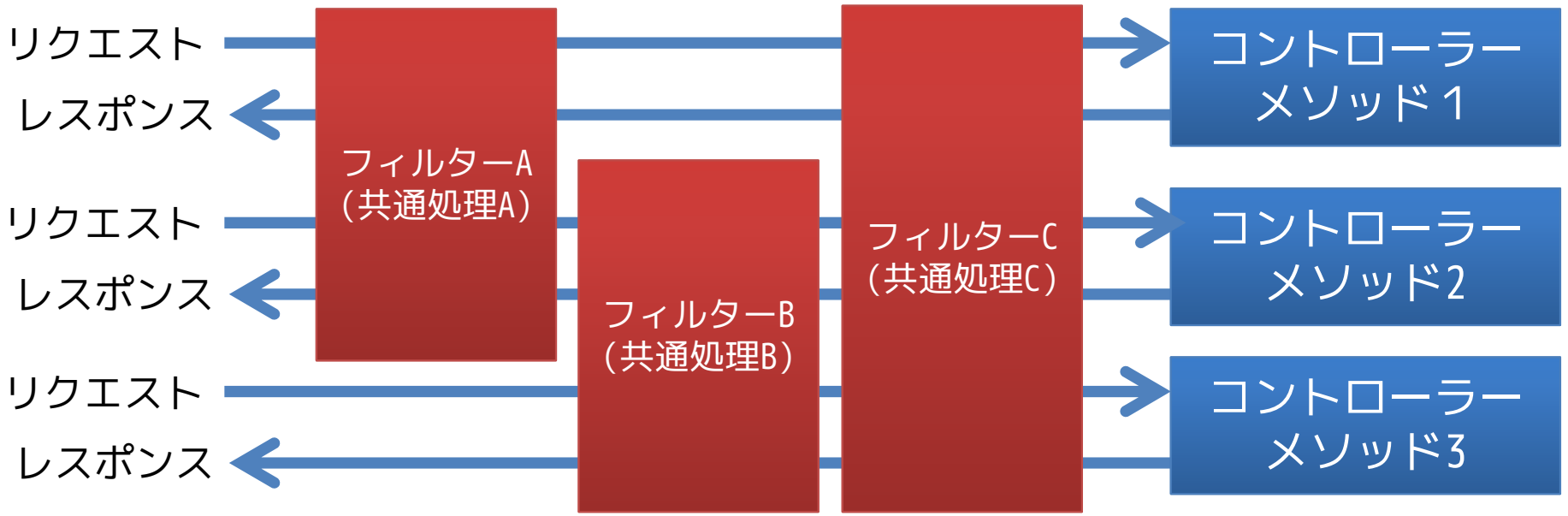
- Webフィルターとは
- Webフィルターの作成と有効化の方法
- Webフィルター内での処理
  - セッションの利用
  - URLの取得
  - リダイレクト

# Webフィルターとは

- Webアプリケーション内で、複数のコントローラーメソッドの共通処理を設定するための仕組み
  - 共通処理：ログイン状態の確認、ログの出力など
  - 静的なリソース(CSSや画像等)のリクエスト時にも適用される

# フィルター・チェーンとは

- フィルターは複数作成することができ「フィルター・チェーン」という仕組みで連結される



# Webフィルターの作成

- jakarta.servlet.Filterインターフェースを実装したクラスとして作成する
  - doFilterメソッドをオーバーライドして実装する

```
public class AuthFilter implements Filter {
    @Override
    public void doFilter(ServletRequest request,
          ServletResponse response,
          FilterChain chain)
                    throws IOException, ServletException {

        chain.doFilter(request, response);
    }
}
```

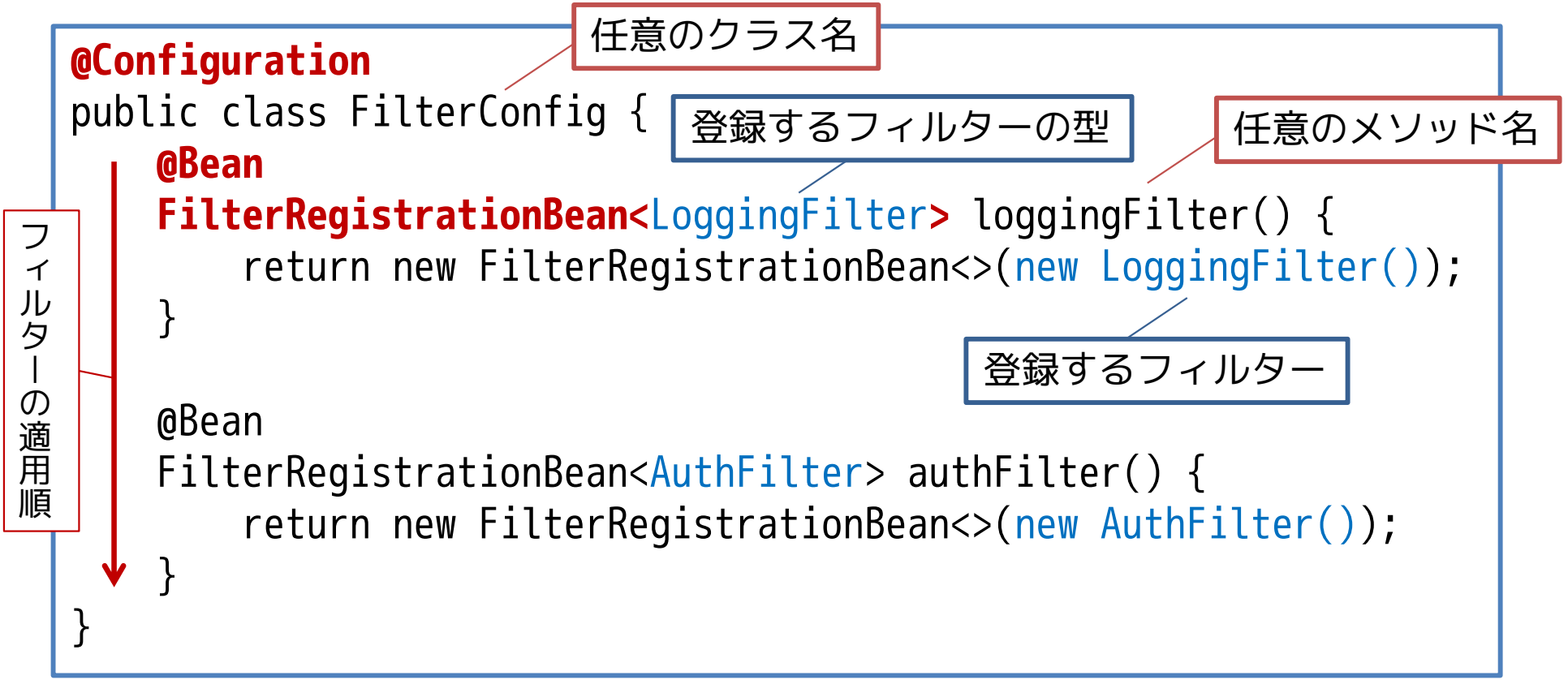
# doFilterメソッド

- doFilterメソッド内では、必ずFilterChain#doFilter()を実行する
  - これによって、次のフィルターに処理が移行する
  - これを挟む形で、リクエスト時の共通処理、レスポンス時の共通処理を記述する

```
@Override
public void doFilter(ServletRequest request,
          ServletResponse response,
          FilterChain chain)
                    throws IOException, ServletException {
        /* リクエスト時の共通処理を記述 */
        chain.doFilter(request, response);
        /* レスポンス時の共通処理を記述 */
}
```

# Webフィルターの有効化

- Webフィルターを有効化するには、設定用クラスファイル(@Configurationを付与したクラス)を作成する
  - FilterRegistrationBeanを返す設定(メソッド)を記述する




```
@Configuration
public class FilterConfig {
    @Bean
    FilterRegistrationBean<LoggingFilter> loggingFilter() {
        return new FilterRegistrationBean<>(new LoggingFilter());
    }

    @Bean
    FilterRegistrationBean<AuthFilter> authFilter() {
        return new FilterRegistrationBean<>(new AuthFilter());
    }
}
```

# Webフィルターの設定

- Webフィルターを登録する際に、対応するURLや適用順などを設定することができる
  - FilterRegistrationBeanのメソッドを使用する

例：AuthFilterを/admins以下のURLに適用する

```
@Bean
FilterRegistrationBean<AuthFilter> authFilter()  {
  var filter = new FilterRegistrationBean<>(new AuthFilter());
  filter.addUrlPatterns("/admin/*"); // * は全てのパターンを示す
  filter.setOrder(1); // 適用順：数値が少ない方が先に適用される
  return filter;
}
```

addUrlPatterns(): 引数は可変長なので、複数のURLを列挙できる
setOrder(): 引数の数値が大きい方が後に適用される

# 練習

- 練習10-1

# doFilterメソッド内での処理

# セッションの利用

- doFilterメソッド内でセッションを利用するには、HttpServletRequest#getSession()を使用する
  - doFilterの引数はServletRequest型なのでキャストが必要

```
@Override
public void doFilter(ServletRequest request,
          ServletResponse response,
          FilterChain chain)
                    throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest) request;
    HttpSession session = req.getSession();
    /* セッションを利用する処理の記述 */
    chain.doFilter(request, response);
}
```

キャストが必要

# HttpServletRequestのメソッド

- HttpServletRequestのおもなメソッド

| メソッド | 説明 |
|---|---|
| String getRequestURI() | http://localhost:8080/member?id=5<br>上記のURLの青字部分を取得する |
| String getQueryString() | 上記のURLの緑字部分を取得する |
| String getParameter(String name) | GET・POSTパラメーターを取得する |
| HttpSession getSession() | セッションを取得する |
| Cookie[] getCookies() | クッキーを取得する |
| void setAttribute(String name,<br>Object o) | Model#addAttributeのように、フォワード先にデータを渡すためのメソッド |
| Object getAttribute(String name) | setAttributeで保存されたデータを取得するためのメソッド |

# リダイレクト

- リダイレクトを実行するには、HttpServletResponse#sendRedirect()を使用する
  - doFilterの引数はServletResponse型なのでキャストが必要

```
@Override
public void doFilter(ServletRequest request,
          ServletResponse response,
          FilterChain chain)
                    throws IOException, ServletException {
    HttpServletResponse res = (HttpServletResponse) response;
    if(ログインしていない場合) {
         res.sendRedirect("/login");
         return;
    }
    chain.doFilter(request, response);
}
```

キャストが必要

# 練習

- 練習10-2

# 補足: OncePerRequestFilter

# OncePerRequestFilter

- Springが提供するフィルタークラスで、ここまでで学習してきたjakarta.servlet.Filterよりも利便性が高い
- 利用時は、doFilterInternalメソッドをオーバーライドして処理を記述する
  - あらかじめ引数がHttpServletRequest, HttpServletResponse型になっている ⇒ jakarta.servlet.Filter#doFilterで必要だった引数のキャストが不要
- リクエストごとに１度だけ実行されることが保証される
  - リクエストがフォワードされる際には実行されない

# OncePerRequestFilterの利用例

例: 認証フィルターの作成

OncePerRequestFilterを継承

doFilterInternalをオーバーライド

引数が使いやすい型になっている

```
public class AuthFilter extends OncePerRequestFilter {
  @Override
  protected void doFilterInternal(HttpServletRequest request,
          HttpServletResponse response, FilterChain filterChain)
                                  throws ServletException, IOException {

    // ログイン済みかチェック
    if(request.getSession().getAttribute("loginInfo") == null) {
        response.sendRedirect("/login");
        return;
    }

    filterChain.doFilter(request, response);
  }
}
```

doFilterメソッドが必須