町内会管理システム  
プログラム設計書

第1.0版

古川 愛

**目次**

[1. 本書の位置付け 4](#_gjdgxs)

[2. 機能仕様 4](#_30j0zll)

[3. ソフトウェア構成 4](#_1fob9te)

[3.1 バックエンド 4](#)

[3.2 フロントエンド 5](#)

[4. ファイル構成 5](#)

[5. クラス構成 7](#_3znysh7)

[5.1 クラス図 7](#_2et92p0)

[5.2 クラス詳細 7](#_tyjcwt)

[5.2.1 Memberクラス 8](#_3dy6vkm)

[5.2.2 Groupクラス 8](#_pet3nqj9vt76)

[このクラスは、上記各フィールドについてアクセッサメソッドを備える。 8](#_ngrl2m8omrf4)

[5.2.3 Adminクラス 8](#_283ltsmmhya1)

[このクラスは、上記各フィールドについてアクセッサメソッドを備える。 9](#_vc7koi533vmg)

[5.2.4 Notificationクラス 9](#_mjv7jldb91wk)

[このクラスは、上記各フィールドについてアクセッサメソッドを備える。 9](#_cdtbs0fcwrhm)

[5.2.5 CircularBoardクラス 9](#_1t3h5sf)

[5.2.6 ReadStatusクラス 10](#_w3vlbijon6ae)

[5.2.7 Scheduleクラス 10](#_1f1u9asacgob)

[5.2.8 ChatMessageクラス 10](#_dq2ms65dyptg)

[5.2.9 Questionnaireクラス 11](#_n461q7xyfqvk)

[5.2.10 ConnectionFileクラス 11](#_jd8y704ijk9f)

[5.2.11 Opinionクラス 11](#_wzhe11c9qivq)

[5.2.12 GenericDao<T>インターフェース 12](#_1wx0vjnizi1y)

[5.2.13 MemberDaoインターフェース 12](#_yk1munv1w76j)

[5.2.14 GroupDaoインターフェース 13](#_h9bk15uat5cz)

[5.2.15 AdminDaoインターフェース 13](#_4d34og8)

[5.2.16 NotificationDaoインターフェース 13](#)

[5.2.17 CircularBoardDaoインターフェース 14](#_hpo6kybzq8pj)

[5.2.18 ReadStatusDaoインターフェース 14](#_7jkbs5fyq4rf)

[5.2.19 ScheduleDaoインターフェース 14](#_l0osk0b4a6er)

[5.2.20 ChatMessageDaoインターフェース 15](#_7gm88c6mekc)

[5.2.21 QuestionnaireDaoインターフェース 15](#_2pwls6wv2sjd)

[5.2.22 ConnectionFileDaoインターフェース 16](#_rp2tbjqds9sg)

[5.2.23 OpinionDaoインターフェース 16](#_vqaaufvf866z)

[5.2.24 GenericService<T>クラス 16](#_1qs0zqcz7g80)

[5.2.25 AdminServiceインターフェース 17](#_xm1r8zgcnqt5)

[5.2.26 AdminServiceImplクラス 17](#_jk6xexr3suzg)

[5.2.27 NotificationServiceインターフェース 17](#_701pkwuk1lue)

[5.2.28 NotificationServiceImplクラス 18](#_d72cc6yufgsn)

[5.2.29 CircularBoardServiceインターフェース 18](#_jex74xs23qm4)

[5.2.30 CircularBoardServiceImplクラス 18](#_gbudj8t9bked)

[5.2.31 ScheduleServiceインターフェース 18](#_eevx402yu4kx)

[5.2.32 ScheduleServiceImplクラス 19](#_y2uarveb3rzl)

[5.2.33 ChatMessageServiceインターフェース 19](#_7a3iyvoft2b9)

[5.2.34 ChatMessageServiceImplクラス 19](#_9wkag6x19srp)

[5.2.35 QuestionnaireServiceインターフェース 20](#_imtn365q8o4f)

[5.2.36 QuestionnaireServiceImplクラス 20](#_tgu9z5x5vsj7)

[5.2.37 ConnectionFileServiceインターフェース 20](#_um852g7amgcw)

[5.2.38 ConnectionFileServiceImplクラス 21](#_1fy542j5xdv)

[5.2.39 OpinionServiceインターフェース 21](#_2sqxvplm8qfw)

[5.2.40 OpinionServiceImplクラス 21](#_iq9cgrflsuen)

[5.2.41 MemberServiceインターフェース 21](#_m7y5jd7pmo18)

[5.2.42 MemberServiceImplクラス 22](#_msq4kkl7utvd)

[5.2.43 LoginControllerクラス 22](#_1ksv4uv)

[5.2.44 RegisterControllerクラス 23](#_ls2toh8zwkc2)

[5.2.45 AdminControllerクラス 24](#_44sinio)

[5.2.46 MemberControllerクラス 28](#_2jxsxqh)

[5.2.47 AdminAuthFilterクラス 29](#_3j2qqm3)

[5.2.48 MemberAuthFilterクラス 30](#_7hsmy7pkwtvw)

[5.2.49 ApplicationConfigクラス 30](#_kz59hi31pl1x)

[6. 改訂履歴 31](#_4i7ojhp)

# 本書の位置付け

本書は、町内会管理システム（以下、「本システム」）に関するプログラムの設計を記す。

# 機能仕様

機能仕様については、以下のドキュメントを参照のこと：

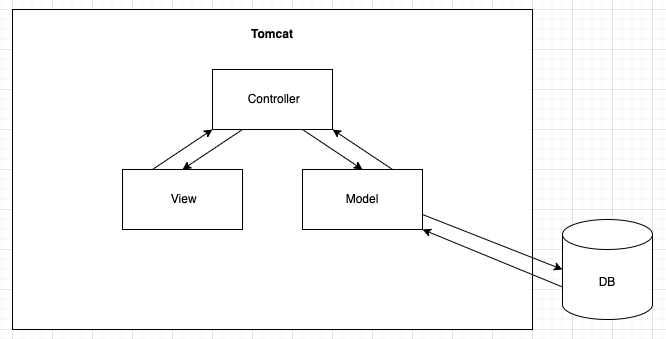
* システム機能一覧
* システム画面設計書
* システムデータ設計書

# ソフトウェア構成

以下に、本システムのソフトウェア構成を示す。

## バックエンド

バックエンドはSpringbootで作成し、データベースはMySQLを使用する。データベース接続にはSpring JDBCを使用する。サーバーはTomcatを使用する。



## フロントエンド

フロントエンドはHTML/CSS/JavaScriptで作成する。ライブラリはBootstrapとjQueryを使用する（下図）。

テーブル が含まれている画像

自動的に生成された説明

# ファイル構成

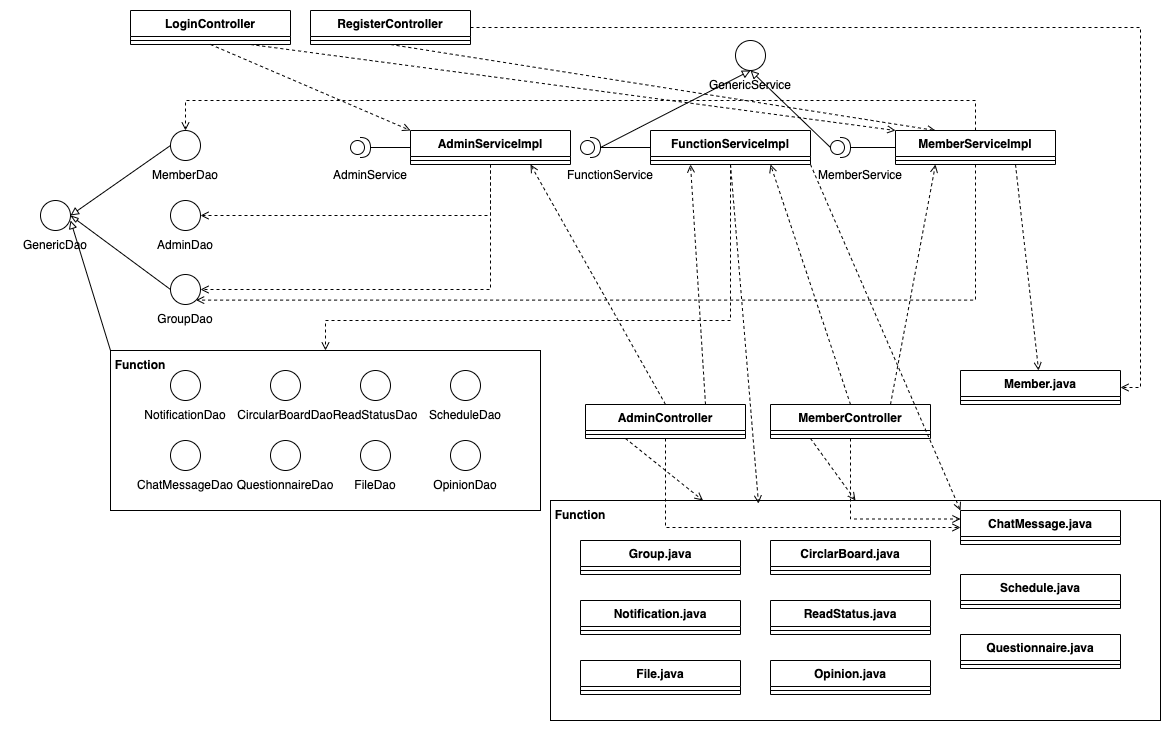
以下に本システムのファイル構成を示す。

| +------ | src/main/java | | | |
| --- | --- | --- | --- | --- |
| | +------ com.example.app | | | | |
| | | +------ | config/ | | |
| | | | | +------ | ApplicationConfig.java | |
| | | +------ | controller/ | | |
| | | | | +------ | LoginController.java | |
| | | | | +------ | RegisterController.java | |
| | | | | +------ | AdminController.java | |
| | | | | +------ | MemberController.java | |
| | | +------ | dao/ | | |
| | | | | +------ | GenericDao<T>.java | |
| | | | | +------ | MemberDao.java | |
| | | | | +------ | GroupDao.java | |
| | | | | +------ | AdminDao.java | |
| | | | | +------ | NotificationDao.java | |
| | | | | +------ | CircularBoardDao.java | |
| | | | | +------ | ReadStatusDao.java | |
| | | | | +------ | ScheduleDao.java | |
| | | | | +------ | ChatMessageDao.java | |
| | | | | +------ | QuestionnaireDao.java | |
| | | | | +------ | ConnectionFileDao.java | |
| | | | | +------ | OpinionDao.java | |
| | | +------ | domain/ | | |
| | | | | +------ | Member.java | |
| | | | | +------ | Group.java | |
| | | | | +------ | Admin.java | |
| | | | | +------ | Notification.java | |
| | | | | +------ | CircularBoard.java | |
| | | | | +------ | ReadStatus.java | |
| | | | | +------ | Schedule.java | |
| | | | | +------ | ChatMessage.java | |
| | | | | +------ | Questionnaire.java | |
| | | | | +------ | ConnectionFile.java | |
| | | | | +------ | Opinion.java | |
| | | +------ | filter/ | | |
| | | | | +------ | AdminAuthFilter.java | |
| | | | | +------ | MemberAuthFilter.java | |
| | | +------ | service/ | | |
| | | | | +------ | GenericService.java | |
| | | | | +------ | AdminService.java | |
| | | | | +------ | AdminServiceImpl.java | |
| | | | | +------ | ChatMessageService.java | |
| | | | | +------ | ChatMessageServiceImpl.java | |
| | | | | +------ | CircularBoardService.java | |
| | | | | +------ | CircularBoardServiceImpl.java | |
| | | | | +------ | ConnectionFileService.java | |
| | | | | +------ | ConnectionFileServiceImpl.java | |
| | | | | +------ | GroupService.java | |
| | | | | +------ | GroupServiceImpl.java | |
| | | | | +------ | MemberService.java | |
| | | | | +------ | MemberServiceImpl.java | |
| | | | | +------ | NotificationService.java | |
| | | | | +------ | NotificationServiceImpl.java | |
| | | | | +------ | OpinionService.java | |
| | | | | +------ | OpinionServiceImpl.java | |
| | | | | +------ | QuestionnaireService.java | |
| | | | | +------ | QuestionnaireServiceImpl.java | |
| | | | | +------ | ScheduleService.java | |
| | | | | +------ | ScheduleServiceImpl.java | |
| +------ | src/main/resources | | | |
| | | +------ | mybatis/ | | |
| | | | | +------ | MemberMapper.xml | |
| | | | | +------ | GroupMapper.xml | |
| | | | | +------ | AdminMapper.xml | |
| | | | | +------ | NotificationMapper.xml | |
| | | | | +------ | CircularBoardMapper.xml | |
| | | | | +------ | ReadStatusMapper.xml | |
| | | | | +------ | ScheduleMapper.xml | |
| | | | | +------ | ChatMessageMapper.xml | |
| | | | | +------ | QuestionnaireMapper.xml | |
| | | | | +------ | FileMapper.xml | |
| | | | | +------ | OpinionMapper.xml | |
| | | +------ | static/ | | |
| | | | | +------ | css/ | |
| | | | | | | +------ | bootstrap.min.css |
| | | | | | | +------ | style.css |
| | | | | +------ | js/ | |
| | | | | | | +------ | bootstrap.bundle.min.js |
| | | | | | | +------ | main.js |
| | | | | | | +------ | jquery-3.7.1.min.js |
| | | | | +------ | images/ | |
| | | | | +------ | docs/ | |
| | | +------ | templates/ | | |
| | | | | +------ | common/ | |
| | | | | +------ | admin/ | |
| | | | | +------ | member/ | |
| | | +------ | application.properties | | |
| | | +------ | validation.properties | | |
| +------ | uploads/ | | | |

# クラス構成

## クラス図

以下に本システムを構成するクラス図を示す。



## クラス詳細

以下に各クラスの詳細を示す。

### Memberクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | 会員ID |
| private String name | 氏名 |
| private String nameKana | 氏名(カナ) |
| private String address | 住所 |
| private String tel | 電話番号 |
| private String email | メールアドレス |
| private Integer familyNumber | 家族人数 |
| private String password | パスワード |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |
| private Date updated | 更新日時 |
| private Group group | グループ情報 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### Groupクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | グループID |
| private String name | グループ名 |
| private Integer deleteFlg | 削除フラグ |

### このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### Adminクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | 管理者ID |
| private String loginId | ログインID |
| private String loginPass | パスワード |
| private String email | メールアドレス |
| private String tel | 電話番号 |

### このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### Notificationクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | お知らせID |
| private String content | お知らせ内容 |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |
| private Date updated | 更新日時 |

### このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### CircularBoardクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | 回覧板ID |
| private String name | 画像名 |
| private String filePath | 画像の保存先パス |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### ReadStatusクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | 既読管理ID |
| private Integer memberId | 会員ID |
| private Integer circularBoardId | 回覧板ID |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### Scheduleクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | スケジュールID |
| private String content | イベント内容 |
| private Date eventAt | 開催日時 |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |
| private Date updated | 更新日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### ChatMessageクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | チャットID |
| private String message | メッセージ内容 |
| private Member member | 会員情報 |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### Questionnaireクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | アンケートID |
| private String name | アンケート名 |
| private String url | アンケートURL |
| private Date deadline | 回答期日 |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### ConnectionFileクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | 資料ID |
| private String name | 画像名 |
| private String note | 備考 |
| private String filePath | 画像の保存先パス |
| private Integer deleteFlg | 削除フラグ |
| private Date created | 登録日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### Opinionクラス

パッケージ： domain

| フィールド | 説明 |
| --- | --- |
| private Integer id | ご意見ID |
| private String content | ご意見内容 |
| private Member member | 会員情報 |
| private Date created | 登録日時 |

このクラスは、上記各フィールドについてアクセッサメソッドを備える。

### 

### GenericDao<T>インターフェース

パッケージ： dao

| メソッド | 説明 |
| --- | --- |
| public Long count() | 有効な数を取得する。  (ページ分割機能用) |
| List<T> selectLimited(@Param("offset") int offset, @Param("num") int num) | 指定したページのオブジェクトのリストを取得する。  int offset: ページ番号  int num: 1ページあたりの件数 |

### MemberDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Member> selectAll() | 有効な会員のリストを取得する。 |
| public Member selectById  (Integer id) | 指定したidの会員を取得する。  Integer id: 会員ID |
| public Member selectByEmail(String email) | 指定したメールアドレスの会員を取得する。 |
| public void insert  (Member member) | 会員情報を保存する。 |
| public void update  (Member member) | 会員情報を更新する。 |
| public void delete  (Member member) | 会員情報を論理削除する。 |

### GroupDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Group> selectAll() | 全グループのリストを取得する。 |
| public Group selectById  (Integer id) | 指定したidのグループを取得する。  Integer id: グループID |
| public void insert  (Group group) | グループ情報を保存する。 |
| public void update  (Group group) | グループ情報を更新する。  (フラグの変更) |

### AdminDaoインターフェース

パッケージ： dao

| メソッド | 説明 |
| --- | --- |
| public Admin selectByLoginId(String loginId) | 指定したログインIDの管理者を取得する。  String loginId: ログインID |
| public Admin selectById(Integer id) | 指定したidの管理者を取得する。  Integer id: 管理者ID |

### NotificationDaoインターフェース

パッケージ： dao　GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Notification> selectAll() | 有効なお知らせのリストを取得する。 |
| public Notification selectById  (Integer id) | 指定したidのお知らせを取得する。  Integer id: お知らせID |
| public void insert  (Notification notification) | お知らせ情報を保存する。 |
| public void update  (Notification notification) | お知らせ情報を更新する。 |
| public void delete  (Notification notification) | お知らせ情報を論理削除する。 |

### CircularBoardDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<CircularBoard> selectAll() | 有効な回覧板のリストを取得する。 |
| public CircularBoard selectById  (Integer id) | 指定したidのお知らせを取得する。  Integer id: 回覧板ID |
| public void insert  (CircularBoard circularBoard) | 回覧板情報を保存する。 |
| public void delete  (CircularBoard circularBoard) | 回覧板情報を論理削除する。 |

### ReadStatusDaoインターフェース

パッケージ： dao

| メソッド | 説明 |
| --- | --- |
| public ReadStatus selectById(Integer id) | 指定したidの既読管理を取得する。  Integer id: 既読管理ID |
| List<ReadStatus> selectBy MemberId(Integer memberId) | 指定した会員idの既読管理を取得する。  Integer memberId: 会員ID |
| public void insert  (ReadStatus readStatus) | 既読管理情報を保存する。 |

### ScheduleDaoインターフェース

パッケージ： dao　GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Schedule> selectAll() | 有効なスケジュールのリストを取得する。 |
| public Schedule selectById  (Integer id) | 指定したidのスケジュールを取得する。  Integer id: スケジュールID |
| public void insert  (Schedule schedule) | スケジュール情報を保存する。 |
| public void update  (Schedule schedule) | スケジュール情報を更新する。 |
| public void delete  (Schedule schedule) | スケジュール情報を論理削除する。 |

### ChatMessageDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<ChatMessage> selectAll() | 有効なチャットのリストを取得する。 |
| public ChatMessage selectById  (Integer id) | 指定したidのチャットを取得する。  Integer id: チャットID |
| public void insert  (ChatMessage chatMessage) | チャット情報を保存する。 |
| public void delete  (ChatMessage chatMessage) | チャット情報を削除する。 |

### QuestionnaireDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Questionnaire> selectAll() | 有効なアンケートのリストを取得する。 |
| public Questionnaire selectById  (Integer id) | 指定したidのアンケートを取得する。  Integer id: アンケートID |
| public void insert  (Questionnaire questionnaire) | アンケート情報を保存する。 |
| public void delete  (Questionnaire questionnaire) | アンケート情報を削除する。 |

### ConnectionFileDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<ConnectionFile> selectAll() | 有効な資料のリストを取得する。 |
| public ConnectionFile selectById  (Integer id) | 指定したidの資料を取得する。  Integer id: 資料ID |
| public void insert  (ConnectionFile connectionFile) | 資料情報を保存する。 |
| public void delete  (ConnectionFile connectionFile) | 資料情報を削除する。 |

### OpinionDaoインターフェース

パッケージ： dao

GenericDaoを継承する。

| メソッド | 説明 |
| --- | --- |
| public List<Opinion> selectAll() | 全ご意見のリストを取得する。 |
| public Opinion selectById  (Integer id) | 指定したidのご意見を取得する。  Integer id: ご意見ID |
| public void insert  (Opinion opinion) | ご意見情報を保存する。 |

### GenericService<T>クラス

パッケージ： service　抽象クラス

| メソッド | 説明 |
| --- | --- |
| protected abstract GenericDao<T> getDao() | 各エンティティ毎のDAOを提供 |
| public int getTotalPages  (int numPerPage) | 有効なトータルページを取得する。 |
| List<T> getListByPage  (int page, int numPerPage) | 指定したページのオブジェクトのリストを取得する。  int page: ページ番号  int numPerPage: 1ページあたりの件数 |

### AdminServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public boolean isCorrectIdAndPassWord  (String loginId, String loginPass) | 管理者のログインIDとパスワードが一致しているかを判定する。  String loginId: 管理者ログインID  String loginPass: 管理者パスワード |
| public Admin getAdminByLoginId  (String loginId) | 指定したログインidの管理者を取得する。  String loginId: 管理者ログインID |

### AdminServiceImplクラス

パッケージ： service

このクラスはAdminServiceインタフェースの実装クラスである。

### NotificationServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<Notification>  getNotificationList() | 有効なお知らせリストを取得する。 |
| public Notification getNotificationById(Integer id) | 指定したidのお知らせを取得する。  Integer id: お知らせID |
| public void addNotification  (Notification notification) | お知らせ情報を登録する。 |
| public void editNotification  (Notification notification) | お知らせ情報を更新する。 |
| public void deleteNotification  (Notification notification) | お知らせ情報を論理削除する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<Notification> getListByPage(int page, int numPerPage) | 指定ページのお知らせリストを取得する。 |

### NotificationServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはNotificationServiceインタフェースの実装クラスである。

### CircularBoardServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<CircularBoard>  getCircularBoardList() | 有効な回覧板リストを取得する。 |
| public CircularBoard getCircularBoardById(Integer id) | 指定したidの回覧板を取得する。  Integer id: 回覧板ID |
| public void addCircularBoard  (CircularBoard circularBoard) | 回覧板情報を登録する。 |
| public void deleteCircularBoard  (CircularBoard circularBoard) | 回覧板情報を論理削除する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<CircularBoard> getListByPage(int page, int numPerPage) | 指定ページの回覧板リストを取得する。 |

### CircularBoardServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはCircularBoardServiceインタフェースの実装クラスである。

### ScheduleServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<Schedule>  getScheduleList() | 有効なスケジュールリストを取得する。 |
| public Schedule getScheduleById(Integer id) | 指定したidのスケジュールを取得する。  Integer id: スケジュールID |
| public void addSchedule  (Schedule schedule) | スケジュール情報を登録する。 |
| public void editSchedule  (Schedule schedule) | スケジュール情報を更新する。 |
| public void deleteSchedule  (Schedule schedule) | スケジュール情報を論理削除する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<Schedule> getListByPage(int page, int numPerPage) | 指定ページのスケジュールリストを取得する。 |

### ScheduleServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはScheduleServiceインタフェースの実装クラスである。

### ChatMessageServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<ChatMessage>  getChatMessageList() | 有効なチャットリストを取得する。 |
| public ChatMessage getChatMessageById(Integer id) | 指定したidのチャットを取得する。  Integer id: チャットID |
| public void addChatMessage  (ChatMessage chatMessage) | チャット情報を登録する。 |
| public void deleteChatMessage  (ChatMessage chatMessage) | チャット情報を論理削除する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<ChatMessage> getListByPage(int page, int numPerPage) | 指定ページのチャットリストを取得する。 |

### ChatMessageServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはChatMessageServiceインタフェースの実装クラスである。

### QuestionnaireServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<Questionnaire>  getQuestionnaireList() | 有効なアンケートリストを取得する。 |
| public Questionnaire getQuestionnaireById(Integer id) | 指定したidのアンケートを取得する。  Integer id: アンケートID |
| public void addQuestionnaire  (Questionnaire questionnaire) | アンケート情報を登録する。 |
| public void deleteQuestionnaire  (Questionnaire questionnaire) | アンケート情報を論理削除する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<Questionnaire> getListByPage(int page, int numPerPage) | 指定ページのアンケートリストを取得する。 |

### QuestionnaireServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはQuestionnaireServiceインタフェースの実装クラスである。

### ConnectionFileServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<ConnectionFile>  getConnectionFileList() | 有効な資料リストを取得する。 |
| public ConnectionFile getConnectionFileById(Integer id) | 指定したidの資料を取得する。  Integer id: 資料ID |
| public void addConnectionFile  (ConnectionFile connectionFile) | 資料情報を登録する。 |
| 資料 | 資料 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<ConnectionFile> getListByPage(int page, int numPerPage) | 指定ページの資料リストを取得する。 |

### ConnectionFileServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはConnectionFileServiceインタフェースの実装クラスである。

### OpinionServiceインターフェース

パッケージ： service

| メソッド | 説明 |
| --- | --- |
| public List<Opinion>  getOpinionList() | 有効なご意見リストを取得する。 |
| public Opinion getOpinionById(Integer id) | 指定したidのご意見を取得する。  Integer id: ご意見ID |
| public void addOpinion  (CircularBoard circularBoard) | ご意見情報を登録する。 |
| public int getTotalPages(int numPerPage) | 有効な総ページ数を取得する。 |
| public List<Opinion> getListByPage(int page, int numPerPage) | 指定ページのご意見リストを取得する。 |

### OpinionServiceImplクラス

パッケージ： service GenericService<T>を継承する。

このクラスはOpinionServiceインタフェースの実装クラスである。

### MemberServiceインターフェース

パッケージ： service

GenericService<T>を継承する。

| メソッド | 説明 |
| --- | --- |
| public boolean isCorrectEmailAndPassWord  (String email, String password) | 会員のメールアドレスとパスワードが一致しているかを判定する。  String email: 会員メールアドレス  String password: 会員パスワード |
| public Member getMemberByEmail  (String email) | 指定したメールアドレスの会員を取得する。  String email: 会員メールアドレス |
| public List<Member>  getMemberList() | 有効な会員リストを取得する。 |
| public Member getMemberById(Integer id) | 指定したidの会員を取得する。  Integer id: 会員ID |
| public void addMember  (Member member) | 会員情報を登録する。 |
| public void editMember  (Member member) | 会員情報を更新する。 |
| public void editPassword  (Member member) | 会員のパスワードを更新する。 |
| public void deleteMember  (Member member) | 会員情報を論理削除する。 |

### MemberServiceImplクラス

パッケージ： service

このクラスはMemberServiceインタフェースの実装クラスである。

### LoginControllerクラス

パッケージ： controller

・管理者用

| メソッド | 説明 |
| --- | --- |
| public String adminLoginGet  (@RequestParam(name = "status", required = false) String status,  Model model) | ​​@GetMapping("/admin/login")  adminLogin.htmlを表示する。 |
| public String adminLoginPost  (@Valid Admin admin, Errors errors, HttpSession session) | @PostMapping(“/admin/login”)  入力の不備をチェックする。  管理者ログインIDと管理者パスワードが正しいかを確認する。  正しい場合、ログインIDをセッションに格納する。 |
| public String adminLogout  (HttpSession session) | @GetMapping("/admin/logout")  セッションを破棄し、管理者ログインページへ遷移。 |
| private String getStatusMessage(String status) | ログアウトメッセージを生成する。  String status: ログアウト時に送られるパラメータ |

・会員用

| メソッド | 説明 |
| --- | --- |
| public String memberLoginGet  (Model model) | ​​@GetMapping("/login")  login.htmlを表示する。 |
| public String memberLoginPost  (@Valid Member member,  Errors errors, HttpSession session) | @PostMapping(“/login”)  入力の不備をチェックする。  会員メールアドレスと会員パスワードが正しいかを確認する。  正しい場合、会員IDと氏名をセッションに格納する。 |
| public String memberLogout  (HttpSession session) | @GetMapping("/logout")  セッションを破棄し、logout.htmlを表示。 |

### RegisterControllerクラス

パッケージ： controller

| メソッド | 説明 |
| --- | --- |
| public String add  (@Valid Member member, Errors errors, Model model) | ​​@PostMapping("/register")  会員情報を受け取り、保存する。 |
| public String editGet  (@PathVariable Integer id, Model model) | @GetMapping(“/edit/{id}”)  memberEdit.htmlを表示する。 |
| public String editPost  (@PathVariable Integer id,  @Valid Member member,  Errors errors, Model model) | @PostMapping(“/edit/{id}”)  指定したidの変更された会員情報を受け取り、保存する。 |
| public String passEditGet  (@PathVariable Integer id, Model model) | @GetMapping(“/passEdit/{id}”)  passEdit.htmlを表示する。 |
| public String passEditPost  (@PathVariable Integer id,  @Valid Member member,  Errors errors, Model model) | @PostMapping(“/passEdit/{id}”)  指定したidの変更されたパスワードを受け取り、保存する。 |

### AdminControllerクラス

パッケージ： controller

| メソッド | 説明 |
| --- | --- |
| public String home  (@RequestParam(name="status", required=false) String status, Model model) | @GetMapping(“/admin/home”)  adminHome.htmlを表示する。  String status: 登録・編集・削除時のステータス |
| public String notificationAddGet  (Model model) | ​​@GetMapping("/admin/infoRegister")  infoAdd.htmlを表示する。 |
| public String notificationAddPost  (@Valid Notification notification, Errors errors, Model model) | ​​@PostMapping("/admin/infoRegister")  お知らせ情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String notificationEditGet  (@PathVariable Integer id, Model model) | ​​@GetMapping("/admin/infoEdit/{id}")  infoEdit.htmlを表示する。 |
| public String notificationEditPost  (@PathVariable Integer id,  @Valid Notification notification, Errors errors, Model model) | ​​@PostMapping("/admin/infoEdit/{id}")  指定idのお知らせ変更情報を受け取り、保存する。  statusにeditを持たせ、adminHome.htmlにリダイレクトする。 |
| public String notificationDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/infoDelete/{id}”)  指定idのお知らせを論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String boardAddGet  (Model model) | ​​@GetMapping("/admin/boardRegister")  boardAdd.htmlを表示する。 |
| public String boradAddPost  (@Valid CirclarBoard circlarBoard, Errors errors, Model model) | ​​@PostMapping("/admin/boardRegister")  回覧板情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String boardDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/boardDelete/{id}”)  指定idの回覧板を論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String groupAddGet  (Model model) | ​​@GetMapping("/admin/groupRegister")  groupAdd.htmlを表示する。 |
| public String groupAddPost  (@Valid Group group,  Errors errors, Model model) | ​​@PostMapping("/admin/groupRegister")  グループ情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String groupShow  (@PathVariable Integer id, Model model) | ​​@GetMapping("/admin/groupShow/{id}")  指定idのgroupShow.htmlを表示する。 |
| public String groupFlgEdit  (@PathVariable Integer id, Model model) | ​​@GetMapping("/admin/groupFlgEdit/{id}")  指定idのグループのフラグを切り替える。 |
| public String memberShow  (@PathVariable Integer id, Model model) | ​​@GetMapping("/admin/memberShow/{id}")  指定idのmemberShow.htmlを表示する。 |
| public String memberDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/memberDelete/{id}”)  指定idの会員を論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String memberGroupEditGet  (@PathVariable Integer id,  Model model) | @GetMapping("/admin/memberGroupEdit/{id}”)  指定idのmemberGroupEdit.htmlを表示する。 |
| public String memberGroupEditPost  (@PathVariable Integer id,  @Valid Member member,  Model model) | @PostMapping("/admin/memberGroupEdit/{id}")  指定idの会員の変更情報を受け取り、保存する。  statusにeditを持たせ、adminHome.htmlにリダイレクトする。 |
| public String memberContact  (@PathVariable Integer memberId,  Model model) | @GetMapping("/admin/memberContact/{memberId}")  スマホの場合：指定idの会員に電話がかかる。  PCの場合：指定idの会員宛のメール画面起動。 |
| public String scheduleAddGet  (Model model) | ​​@GetMapping("/admin/scheduleRegister")  scheduleAdd.htmlを表示する。 |
| public String scheduleAddPost  (@Valid Schedule schedule, Errors errors, Model model) | ​​@PostMapping("/admin/scheduleRegister")  スケジュール情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String scheduleEditGet  (Model model) | ​​@GetMapping("/admin/scheduleEdit/{id}")  scheduleEdit.htmlを表示する。 |
| public String scheduleEditPost  (@Valid Schedule schedule, Errors errors, Model model) | ​​@PostMapping("/admin/scheduleEdit/{id}")  指定idのスケジュール変更情報を受け取り、保存する。  statusにeditを持たせ、adminHome.htmlにリダイレクトする。 |
| public String scheduleDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/scheduleDelete/{id}”)  指定idのスケジュールを論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String chatDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/chatDelete/{id}”)  指定idのチャットを論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String questionAddGet  (Model model) | ​​@GetMapping("/admin/questionRegister")  boardAdd.htmlを表示する。 |
| public String questionAddPost  (@Valid Questionnaire questionnaire, Errors errors, Model model) | ​​@PostMapping("/admin/questionRegister")  アンケート情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String questionDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/questionDelete/{id}”)  指定idのアンケートを論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| public String fileAddGet  (Model model) | ​​@GetMapping("/admin/fileRegister")  fileAdd.htmlを表示する。 |
| public String fileAddPost  (@Valid ConnectionFile connectionFile, Errors errors, Model model) | ​​@PostMapping("/admin/fileRegister")  資料情報を受け取り、保存する。  statusにaddを持たせ、adminHome.htmlにリダイレクトする。 |
| public String fileDelete  (@PathVariable Integer id, Model model) | @GetMappint(“/admin/fileDelete/{id}”)  指定idの資料を論理削除する。  statusにdeleteを持たせ、adminHome.htmlにリダイレクトする。 |
| private String getStatusMessage(String status) | メッセージを生成する。  String status: パラメータ |

### MemberControllerクラス

パッケージ： controller

| メソッド | 説明 |
| --- | --- |
| public String home  (Model model) | @GetMapping(“/home”)  home.htmlを表示する。 |
| public String board  (Model model) | @GetMapping(“/board”)  board.htmlを表示する。  既読情報を取得して既読/未読の表示をする。 |
| public String boardRead  (@PathVariable Integer boardId,  @PathVariable Integer memberId, Model model) | @GetMapping(“/board/{boardId}/{memberId}”)  既読管理に回覧板IDと会員IDを登録する。 |
| public String prevBoard  (@PathVariable Integer nowMonth, Model model) | @GetMapping(“/prevBoard/{nowMonth}”)  prevBoard.htmlを表示する。  選択された月の回覧板を一覧で表示する。 |
| public String schedule  (@PathVariable Integer nowMonth,  Model model) | @GetMapping(“/schedule/{nowMonth}”)  schedule.htmlを表示する。  選択されたカレンダーの月のスケジュールを一覧で表示する。 |
| public String chat(Model model) | @GetMapping(“/chat”)  chat.htmlを表示する。  バッチ処理：3分毎に画面リロード |
| public String chatPost  (Model model) | @PostMapping(“/chat”)  チャットメッセージを受け取り、保存する。  chat.htmlをリダイレクト。 |
| public String file(Model model) | @GetMapping(“/file”)  file.htmlを表示する。 |
| public String question(Model model) | @GetMapping(“/question”)  question.htmlを表示する。 |
| public String opinion(Model model) | @GetMapping(“/opinion”)  opinionbox.htmlを表示する。 |
| public String opinionPost  (Model model) | @PostMapping(“/opinion”)  ご意見内容を受け取り、保存する。  opinion.htmlをリダイレクト。 |
| public String contact(Model model) | @GetMapping(“/contact”)  contact.htmlを表示する。 |
| public String contactDo  (@PathVariable Integer adminId,  Model model) | @GetMapping(“/contact/{adminId}”)  スマホの場合：指定の管理者IDに電話がかかる。  PCの場合：指定の管理者ID宛のメール画面起動。 |

### AdminAuthFilterクラス

パッケージ： filter

| メソッド | 説明 |
| --- | --- |
| public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException | セッションにログインIDが保存されているかどうかを確認し、保存されていない場合はログイン未認証とみなし、管理者ログイン画面（SK01-1）へリダイレクトする。ただし、管理者ログイン画面（SK01-1）がリクエストされた際にはこのチェックは行わないものとする。 |

### MemberAuthFilterクラス

パッケージ： filter

| メソッド | 説明 |
| --- | --- |
| public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException | セッションに会員IDが保存されているかどうかを確認し、保存されていない場合はログイン未認証とみなし、ログイン画面（SM01-1）へリダイレクトする。ただし、ログイン画面（SM01-1）がリクエストされた際にはこのチェックは行わないものとする。 |

### ApplicationConfigクラス

パッケージ： config

| メソッド | 説明 |
| --- | --- |
| public Validator getValidator()IOException, ServletException | エラーメッセージ用の設定 |
| public ResourceBundleMessageSource messageSource() | エラーメッセージ用の設定 |
| public FilterRegistrationBean<AdminAuthFilter> authFilter() | 管理者認証フィルタの有効化 |
| public FilterRegistrationBean<MemberAuthFilter> studentAuthFilter() | 会員認証フィルタの有効化 |
| public void addResourceHandlers(ResourceHandlerRegistry registry) | uploadsフォルダ設定 |

# 改訂履歴

| 版 | 日付 | 改訂者 | 内容 |
| --- | --- | --- | --- |
| 1.0 | 2024/12/20 | 古川　愛 | 初版作成 |
| 1.1 | 2024/12/25 | 古川　愛 | GenericDao、GenericServiceを追加 |