開発エンジニア面談時テスト

日付　　2024/03/17

所属

お名前　森田美紀

問１. HTTPリクエストのGETメソッドとPOSTメソッドの違いを簡潔にご回答ください。

【回答】

＜GETメソッド＞

・データの取得を目的としたメソッド。

・データをサーバーに送信する際、URLにクエリ文字列として設定する。

・送信されるデータはURLに直接表示される。

＜POSTメソッド＞

・主にデータの登録や更新などの書き込みを目的としたメソッド。

・データをサーバーに送信する際、リクエストボディに設定する。

・送信されるデータはクエストボディに埋め込まれるため、URLに直接表示されない。

開発エンジニア面談時テスト

問２. 会員制Webサイトでセッション・ハイジャックの脆弱性が発見され、対策を実施することになりました。なお、このWebサイトの認証及び認可の仕組みは下記の通りとなっています。

- セッションはCookieで管理されています。

- Webサイトは、非会員でも閲覧可能なページと、会員専用のページとで構成されています。

- 会員専用ページへのアクセスには、必ずログインが求められます。

- ログインの状態はセッションで管理されています。

このWebサイトでセッション・ハイジャック対策として必要となる対応を、簡潔にご回答ください。

【回答】

・セッションIDの盗聴を防ぐため、CookieでSecure属性を付与し、HTTPS接続のみでアクセスできるようにする。

・JavaScriptを利用したセッションIDの盗聴を防ぐため、CookieでHttpOnly属性を付与し、JavaScriptからのアクセスを制限する。

・Webサーバーが生成するセッションIDの生成ルールがある規則に基づいたものである場合、総当たり攻撃の可能性があるため、予測しにくいランダムな値に設定するようにする。

開発エンジニア面談時テスト

問３. ２種類のIDに基づき処理を分岐するという要件に基づき、オブジェクト指向言語を用いて下記の実装が行われました。

if（ID = = "ID-A") {

// ID-A 用の複雑な処理

:

:

｝else if（ID = = "ID-B") {

// ID-B 用の複雑な処理

:

:

｝else｛

// エラー処理

:

:

}

ところがその後、仕様変更となり、新しいIDが複数件追加されることになりました。

また今後も随時、ID追加が発生する見込みです。

1）このプログラムをそのまま踏襲して改修を進めた場合、どのような問題があり得ますか。

2）このプログラムをオブジェクト指向に基づき改善する場合、どのようにすれば良いでしょうか。

それぞれ、簡潔にご回答ください。

【回答】

1）

新しいIDが追加されるたびに条件分岐のコードを変更する必要があるため、保守性が低下する。また、コードの肥大化により可読性が低下する。

2）

条件分岐はif文からcase文に修正し、それぞれのID用の複雑な処理はメソッド化することで可読性の低下を防ぐ。

※もっといい方法があると思いますが、私が考えつくのはこのレベルです。

　可読性については好みがあると思うので、可動性が上がったかというと微妙かもしれません。

　保守性の観点としての対策は思いつきませんでした。

開発エンジニア面談時テスト

問４. ある文字列が「重複する文字を含むか否か」を判断するプログラムを記述してください。

たとえばそのプログラムを使って「abcdde」という文字列を処理した場合は、「重複がある」と判定され、「abcde」という文字列を処理した場合は、「重複がない」と判定されます。

ただし対象とする文字列はASCII文字列のみとします。

また、回答として記述をおこなうプログラム言語は、実在するプログラム言語ではなく疑似言語でもかまいません。

【回答】

/\*\*

\* 重複文字判定処理

\* @param 対象文字列

\* @return true:重複文字あり、false:重複文字なし

\*/

private boolean hasDuplicateCharacters(String inputStr) {

// 文字の重複を判定するためのセット

Set<Character> charSet = new HashSet<>();

// 対象文字列を一文字ずつチェック

for (char c : inputStr.toCharArray()) {

// 文字がセットに既に存在する場合、重複あり

if (charSet.contains(c)) {

return true;

}

// 文字をセットに追加

charSet.add(c);

}

return false;

}