Валидация пользовательских данных

№ урока: 3 **Курс:** Flask

Средства обучения: Python3 и любимая среда разработки, например, PyCharm.

Обзор, цель и назначение урока

Узнать, зачем нужна валидация данных на сервере, если она уже есть на клиентской части, обсудим, что такое сериализация и десериализация данных. Улучшим структуру приложения.

Изучив материал данного занятия, учащийся сможет:

- Валидировать пользовательские данные.
- Использовать сериализацию и десериализацию.
- Создавать правильную структуру для крупного или растущего проекта.

Содержание урока

1. Зачем нужна валидация пользовательских данных на сервере
2. Marshmallow
3. Улучшенная структура проекта

Резюме

- **Что такое валидация формы? -** Откройте любой популярный сайт с формой регистрации, и вы заметите, что они дают вам обратную связь, когда вы вводите ваши данные не в том формате, который они ожидают от вас. Вы получите подобные сообщения:
    - "Это поле обязательно для заполнения" (вы не можете оставить это поле пустым).
    - "Пожалуйста введите ваш телефонный номер в формате xxx-xxxx" (вводит три цифры, разделенные тире, за ними следуют четыре цифры).
    - "Пожалуйста введите настоящий адрес электронной почты" (если ваша запись не в формате"somebody@example.com").
    - "Ваш пароль должен быть от 8 до 30 символов длиной, и содержать одну заглавную букву, один символ, и число".

  Это называется **валидация формы** — когда вы вводите данные, веб-приложение проверяет, что данные корректны. Если данные верны, приложение позволяет данным быть отправленными на сервер и (как правило) быть сохраненными в базе данных; если нет - оно выдает вам сообщение об ошибке, объясняющее какие исправления, необходимо внести. Проверка формы может быть реализована несколькими различными способами. Мы хотим сделать заполнение веб-форм максимально простым.

- Итак, почему мы настаиваем на подтверждении наших форм?
  Существуют три основные причины:
    - Мы хотим получить нужные данные в нужном формате — наши приложения не будут работать должным образом, если данные наших пользователей хранятся в неправильном формате, если они вводят неправильную информацию или вообще не передают информацию.
    - Мы хотим защитить учетные записи наших пользователей, заставляя наших пользователей вводить защищенные пароли. Это упрощает защиту информации об их учетной записи.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Мы хотим обезопасить себя — существует множество способов, которыми злоумышленники могут злоупотреблять незащищенными формами, чтобы повредить приложение, в которое они входят.

- Существует два разных типа проверки формы, с которыми вы столкнетесь в Интернете:
  - Проверка на стороне клиента.
  - Проверка на стороне сервера.

- **Проверка на стороне клиента** — это проверка, которая происходит в браузере, прежде чем данные будут отправлены на сервер. Это удобнее, чем проверка на стороне сервера, так как дает мгновенный ответ. Ее можно далее подразделить на:
  - JavaScript проверка выполняется с использованием JavaScript. Полностью настраиваемая.
  - Встроенная проверка формы, используя функции проверки формы HTML5. Для этого обычно не требуется JavaScript. Встроенная проверка формы имеет лучшую производительность, но она не такая настраиваемая, как с использованием JavaScript.

- **Проверка на стороне сервера** — это проверка, которая возникает на сервере после отправки данных. Серверный код используется для проверки данных перед их сохранением в базе данных. Если данные не проходят проверку валидности, ответ отправляется обратно клиенту, чтобы сообщить пользователю, какие исправления должны быть сделаны. Проверка на стороне сервера не такая удобная, как проверка на стороне клиента, поскольку она не выдает ошибок до тех пор, пока не будет отправлена вся форма. Тем не менее, проверка на стороне сервера — это последняя линия защиты вашего приложения от неправильных или даже вредоносных данных. Все популярные серверные фреймворки имеют функции для проверки и очистки данных (что делает их безопасными).
  В реальном мире разработчики склонны использовать комбинацию проверки на стороне клиента и сервера.

- **Зачем нужна валидация на стороне сервера?** – потому что валидацию на стороне клиента можно обойти, например, использовать Postman или curl, и данные будут отправлены непосредственно на сервер. Например, в Интернете, если вы используете для проверки JavaScript, отключить JavaScript очень просто.
  Вообще, принцип «**никогда не доверяй клиенту**» — это причина, по которой нужно всегда проверять данные на сервере. Вы можете спросить в таком случае, зачем проверять на стороне клиента? - Для того, чтобы обеспечить немедленную обратную связь.

- **Сериализация** — это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл. Эта операция предназначена для того, чтобы сохранить состояния объекта для последующего воссоздания при необходимости. Обратный процесс называется десериализацией.

- **JSON** — один из популярных форматов для сериализации, он текстовый, легковесный и легко читается человеком.

Закрепление материала

- Что такое валидация данных?
- Зачем нужна валидация данных на стороне сервера?
- Зачем нужна валидация данных на стороне клиента?

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Что такое сериализация и десериализация данных?
- Зачем использовать сериализацию данных?

Дополнительное задание

Задание

Создайте схему для валидации модели пользователя по шаблону Film или Actor. Создайте такие поля: username, email и password, используя Marshmallow. Для этих полей определите следующие валидаторы: fields.String и fields.Email, которые должны быть обязательными. В дальнейшем мы будем использовать эту схему валидации для авторизации пользователя.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Изучите материалы из рекомендуемых ресурсов.

Задание 3
Рассмотрите альтернативы Marshmallow: Cerberus, Voluptuous, Pydantic.

Рекомендуемые ресурсы

Валидация данных:
https://developer.mozilla.org/ru/docs/Learn/HTML/Forms/%D0%92%D0%B0%D0%BB%D0%B8%D0%B4%D0%B0%D1%86%D0%B8%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D1%8B

Сериализация данных:
https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/serialization/

Структура проекта:
https://flask.palletsprojects.com/en/1.1.x/tutorial/layout/

https://flask.palletsprojects.com/en/1.1.x/tutorial/factory/

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com