stackoverflow.com

Qt Script function wrapping

You are already on the right track. `QScriptEngine::newFunction()` brings the function into the engine. Now, you need a way to access this function from the script. A "function" is just a property of the global object and you can add a new property with `setProperty()`. The code

```
QScriptValue globalObject = engine.globalObject();
QScriptValue func =
engine.newFunction(returnProperty);
globalObject.setProperty("foo", func);
```

produces the output

```
3
running returnValues Function  "name"
```

```
This prints values from addValues function : name
```

The flags `QScriptValue::PropertyGetter` and `QScriptValue::PropertySetter` are only needed, when you want to create a property, which has to call a function upon access. It is similar to the properties of `QObject`. Consider this example:

```
class MyObject : public QObject
{
   Q_PROPERTY(QString name READ getName WRITE
setName)
};
MyObject* obj = new MyObject;
```

When you do a `obj->setProperty("name", "Sam");` you call `MyObject::setName("Sam")` in the background and `obj->getProperty("name")` is a wrapper for `MyObject::getName()`. A small example:

```
QScriptValue getName(QScriptContext* ctx,
QScriptEngine* eng)
```

```
{
     // Return the value of the internal '_name_'
property.
     qDebug() << "Getter 'getName' called";
     return ctx->thisObject().property("_name_");
}

QScriptValue setName(QScriptContext* ctx,
QScriptEngine* eng)
{
    // Do some processing and store the name in
an internal '_name_' property.
    qDebug() << "Setter 'setName' called";
    ctx->thisObject().setProperty("_name_",

ctx->argument(0).toString().toUpper());
    return QScriptValue::UndefinedValue;
}

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QScriptEngine engine;
    QScriptValue globalObject =
engine.globalObject();

    // Create a new object.
    QScriptValue obj = engine.newObject();
    // Bring the functions into the engine.
    QScriptValue getNameFunc =
engine.newFunction(getName);
    QScriptValue setNameFunc =
engine.newFunction(setName);
    // Create a 'name' property, which calls the
getter and setter from above.
    obj.setProperty("name", getNameFunc,
QScriptValue::PropertyGetter);
    obj.setProperty("name", setNameFunc,
QScriptValue::PropertySetter);
    // Make the new object known as 'person'.
    globalObject.setProperty("person", obj);
    // Test our construct.
    engine.evaluate("print('Set the name to
fitzgerald');");
    engine.evaluate("person.name =
'fitzgerald';");
    engine.evaluate("print('And the name is... '
+ person.name)");
}
```

Finally the output:

```
Set the name to fitzgerald
Setter 'setName' called
Getter 'getName' called
And the name is... FITZGERALD
```